Un système interactif pour le prototypage virtuel coopératif

Olivier Balet

N° Ordre : 1998

Thèse

Présentée devant

l'Université Paul Sabatier de Toulouse (Sciences)

En vue de l'obtention

Du Grade de Docteur de l'Université Paul Sabatier

Spécialité : Informatique

Par Olivier BALET

Un système interactif pour le prototypage virtuel coopératif

Soutenue le 11 juin 1998 devant un jury composé de :

Président :

M.	*R. Caubet*	*Professeur, Université Paul Sabatier, FR*

Rapporteurs :

M.	*P. Coiffet*	*Directeur de recherche CNRS, Laboratoire de Robotique de Pari, FR*
M.	*E. Gobbetti*	*Directeur de recherche, Visualisation & VR Department, CRS4, IT*

Examinateurs :

Mme	*P. Gelband*	*Executive Vice President, Sense8 Corporation, US*
M.	*G. Fisse*	*Responsable Activité Imagerie, CISI, FR*
M.	*Y. Duthen*	*Professeur, Université des Sciences Sociales de Toulouse, FR*
M.	*J.P. Jessel*	*Maître de conférence, Université Paul Sabatier, FR*

Remerciements

Mes plus sincères remerciements vont tout d'abord à Monsieur le professeur René Caubet, pour m'avoir accueilli au sein de l'équipe Synthèse d'Images de l'IRIT et avoir dirigé mes recherches pendant toutes mes années de DEA et de thèse. Je tiens à associer à ces remerciements Messieurs Patrice Berranger et Georges Fisse, respectivement directeur d'agence et responsable de l'activité Imagerie de la société CISI, qui ont su m'accueillir au sein de leur société dans des conditions exceptionnelles. Ces remerciements vont enfin à l'Agence Nationale de la Recherche Technique pour avoir participé au financement des travaux présentés dans ce mémoire.

Pour avoir accepté d'être rapporteurs de cette thèse et avoir eu l'amabilité de participer à son jury, j'exprime mes vifs remerciements à Monsieur Philippe Coiffet, directeur de recherche CNRS au Laboratoire de Robotique de Paris, et à Monsieur Enrico Gobbetti, directeur de recherche du département Visualisation & Réalité Virtuelle du Centre for Advanced Studies, Research and Development in Sardinia (CRS4).

Je remercie également Mme Patrice Gelband, vice présidente et co-fondatrice de la société Sense8, ainsi que Messieurs Jean-Pierre Jessel et Yves Duthen, respectivement professeur à l'université des Sciences Sociales de Toulouse et maître de conférences à l'université Paul Sabatier, pour avoir accepté de participer à ce jury.

Un grand merci à tous les membres du département Réalité Virtuelle de CISI avec qui il est si agréable de travailler, en espérant encore longtemps partager avec eux toutes les journées qu'illuminent leur gentillesse et leur bonne humeur.

Merci à tous les membres de l'équipe Synthèse d'Image de l'IRIT et tout particulièrement à ceux d'entre eux avec qui j'ai eu le plaisir de collaborer et de partager idées et points de vue.

Pour terminer, je remercie ma famille et mes amis, les premiers pour m'avoir donner la force et l'amour nécessaires à l'aboutissement de mes études, les deuxièmes pour m'avoir supporté durant celles qui compteront sans doute comme les plus belles années de ma vie.

Merci donc à tous pour m'avoir permis de mener à bien ce travail dans de telles conditions.

Table des matières

Introduction

Depuis leur apparition, les ordinateurs ont radicalement transformé le travail de l'ingénieur. L'informatique fait désormais partie intégrante de sa formation. Elle est aussi bien utilisée pour concevoir, modéliser et simuler des systèmes complexes que pour communiquer. L'ordinateur a ainsi permis aux designers de se séparer de leurs crayons, gommes, règles et papier pour travailler, souris ou stylet en main, sur des projets visualisés en trois dimensions sur leur écran. Cette révolution ne s'est pourtant pas faite sans douleur. Il suffit, pour s'en convaincre, de rencontrer un de ces pionniers des systèmes de conception assistée par ordinateurs (CAO), un de ceux qui, au milieu des années 60 [Sutherland63] [Ross63], devait passer pour l'original de l'équipe lorsqu'il tentait de convaincre ses collègues de l'intérêt de la machine informatique pour la conception de maquettes numériques. L'industrie était réticente ; les ingénieurs résistaient à ce qui allait révolutionner leur manière de travailler.

Et pourtant, quel bureau d'études pourrait encore se passer des maquettes numériques ? Quel projet dans des industries aussi variées que l'automobile, l'aéronautique, le spatial ou encore le bâtiment pourraient être menés à bout sans utiliser de tels systèmes ?

L'informatique a donc offert aux designers de puissants outils permettant de créer des objets, de les déplacer, les orienter et les assembler les uns avec les autres afin de constituer des systèmes complexes. Pourtant, une limitation de ces systèmes reste liée à la représentation bidimensionnelle de l'environnement de travail proposé à l'utilisateur. Par exemple, orienter un objet dans l'espace avec un système de CAO reste encore une opération peu intuitive, par trop éloignée de celle qu'aurait réalisée l'utilisateur s'il avait eu l'objet en main.

Figure 1 : *Maquette réelle ou maquette virtuelle* [1] *?*

L'introduction des technologies regroupées sous le nom de Réalité Virtuelle est à même de relever le défi d'une interaction homme machine transparente et de permettre la manipulation de manière naturelle d'objets dits virtuels.

L'environnement de travail proposé par les ordinateurs a évolué, au fil de l'histoire, du monde linéaire des terminaux de type VT100 au monde bidimensionnel et coloré de Windows et X-Windows, proposés sur des stations de travail toujours plus puissantes. La technologie semble désormais prête, tant au niveau matériel que logiciel, pour franchir le pas de la 2D et ouvrir l'ère des environnements de travail tridimensionnels. Il est d'ailleurs intéressant de voir combien les réticences semblent resurgir lorsque l'on propose aux ingénieurs une nouvelle évolution dans leurs habitudes de travail. L'homme serait-il par nature conservateur ?

Nous présentons dans ce mémoire l'étude et la réalisation d'un système interactif pour le prototypage coopératif de maquettes virtuelles. Ces travaux font usage de plusieurs technologies issues de milieux scientifiques variés ; la réalité virtuelle n'est elle pas à la croisée des chemins de nombreuses disciplines ?

Notre objectif n'est pas de remplacer dès à présent un système de CAO par un système tel que celui que nous proposons dans ce mémoire. En effet, la puissance des machines ne permet pas encore la gestion d'objets virtuels avec une précision comparable à celle des outils de CAO. Certes notre système est intuitif et interactif mais il ne dispose pas d'assez de puissance machine pour rivaliser en précision avec de tels outils ; cette précision est pourtant nécessaire pour l'industrie. Cette évolution se fera, c'est sûr, mais il est pour l'instant plus raisonnable de voir la réalité virtuelle comme un complément de la CAO.

[1] avec l'aimable autorisation de la société REM Infográfica

Problématique

Les décisions prises au cours des phases de spécification et de conception des projets d'ingénierie à grande échelle sont souvent extrêmement importantes de par leur fort impact sur les résultats finaux ainsi que sur les délais et les coûts de développement. Bien que bénéficiant d'outils informatiques puissants, tels que les systèmes de CAO, la plupart de ces projets nécessitent encore de nos jours la construction de maquettes physiques dont le coût et le manque de flexibilité ont conduit les industriels concernés à étudier de nouvelles solutions.

En effet, la trop grande limitation, en termes de visualisation et d'interaction, des logiciels de CAO actuels rend longues et fastidieuses des opérations aussi nécessaires que l'inspection ou la manipulation interactive des modèles étudiés. De plus, de tels logiciels restent encore réservés à des spécialistes parfois peu impliqués dans les phases d'avant projet où se dessinent et se décident les choix technologiques.

C'est pourquoi, à l'heure du tout informatique, les décideurs et les concepteurs de systèmes complexes sont parfois obligés de construire plusieurs maquettes physiques, avec parfois des moyens de fortune (carton, papier, bouchons en liège), afin d'appréhender et de vérifier, par exemple, la bonne intégration des différents composants ou leur accessibilité une fois installés. De telles maquettes sont utilisées quasi systématiquement dans des domaines aussi variés que l'industrie du spatial, de l'automobile ou de l'architecture.

Parmi les solutions envisagées pour s'affranchir de la construction de maquettes physiques, celle prônant l'utilisation des technologies regroupées sous le nom de Réalité Virtuelle semble répondre le mieux aux besoins exprimés. Remplacer tout ou partie des maquettes réelles par des maquettes virtuelles paraît être une évolution logique de l'informatisation du métier de l'ingénieur. Toutefois, il existe encore un fossé extrêmement large séparant les systèmes de Réalité Virtuelle étudiés dans les laboratoires de recherche des outils industriels attendus.

Quelques rares tentatives [Ellis96] ont pourtant eu lieu afin d'utiliser de tels systèmes au cours de projets industriels. Cependant, les solutions choisies étaient pratiquement toutes limitées à la visualisation de maquettes virtuelles.

Plan du mémoire

Nous présentons, dans ce mémoire, le résultat des travaux que nous avons menés dans le cadre du projet de recherche PROVIS (Prototypage Virtuel de Systèmes), initié en 1994 par le Centre National d'Etudes Spatiales. Ce projet vise à étudier de nouvelles solutions pour simplifier, en premier lieu, le travail des décideurs et des concepteurs de systèmes qui, dès les phases d'avant projet, doivent disposer d'outils leur permettant d'appréhender et de vérifier intuitivement la pertinence de leurs choix. Dans un deuxième temps, de telles solutions se doivent de servir de support à la communication entre les différentes équipes impliquées dans le développement des projets de grande envergure. La dispersion géographique de ces équipes ainsi que la difficulté de communication, inhérente aux différents corps de métier impliqués, nous ont guidé vers une solution faisant usage des techniques de visualisation et d'interaction tridimensionnelles en environnement distribué.

Ce choix fut aussi justifié par la nouvelle organisation du travail que cherchent à mettre en place les grands industriels. Elle se réfère au concept d'ingénierie concourante qui a pour objectif de prendre en compte dès la conception les contraintes des métiers qui interviennent dans le cycle de vie du produit comme, par exemple, les contraintes de fabrication, de maintenance ou de recyclage. Ce type d'organisation du travail vise aussi à faire se chevaucher les différentes étapes d'un projet, nécessitant par là même un support de communication entre les différentes parties impliquées. Le lecteur intéressé par de plus amples renseignements sur cette méthodologie de travail pourra se référer à l'excellent article de Vincent Giard [Giard96].

Nous présenterons un bref historique de la réalité virtuelle et des technologies de l'interaction homme machine dans le premier chapitre de ce mémoire

Nous poursuivrons, dans le second chapitre, en détaillant les différentes méthodes et techniques disponibles pour la modélisation des environnements virtuels. Nous y exposerons les modèles de description géométrique les plus couramment utilisés ainsi que les méthodes disponibles pour construire un modèle virtuel à partir d'un modèle réel. Nous conclurons ce chapitre en présentant quelques approches permettant de simplifier et de rendre plus intuitive la longue et fastidieuse tâche que constitue la modélisation de scènes tridimensionnelles.

Le troisième chapitre détaillera les techniques de prototypage virtuel interactif. Nous présenterons, en premier lieu, les différents types de périphériques disponibles pour l'interaction homme machine tridimensionnelle. Nous poursuivrons en présentant comment ces dispositifs ont pu être utilisés pour accomplir des tâches de modélisation. Nous décrirons ensuite l'utilisation des techniques de prototypage virtuel interactif dans des projets industriels à grande échelle. Enfin, nous présenterons le projet PROVIS, cadre des travaux présentés dans ce mémoire.

Nous détaillerons, dans le quatrième chapitre, la couche logicielle VIPER que nous proposons pour le développement d'applications de réalité virtuelle distribuée. Nous en présenterons la structure logique adaptée à la conception d'une application coopérative. Nous proposerons un modèle comportemental permettant de définir de manière souple et robuste le comportement des objets virtuels ainsi que leurs interactions. Enfin, nous présenterons une évaluation des différentes techniques pour la gestion des communications audio et vidéo entre les différents utilisateurs qui partagent, à distance, une même session de travail.

Le chapitre cinq présentera l'architecture du système PROVIS et montrera comment ce système s'articule au dessus de la plate-forme logicielle VIPER. Nous détaillerons les différentes métaphores d'interaction utilisées pour communiquer avec les autres utilisateurs ou manipuler et assembler les composants du prototype virtuel. Nous définirons ensuite les mécanismes mis en œuvre pour supporter le dialogue multimodal entre l'homme et la machine. Enfin, nous proposerons l'application du modèle comportemental, défini au chapitre précédent, pour la création de mécanismes composés (targette, levier de vitesse, etc.).

Nous poursuivrons, chapitre six, en nous intéressant à l'utilisation d'une méthode originale, basée sur une approche évolutionniste, pour la définition de comportements pour objets articulés. Les méthodes évolutionnistes permettent de résoudre un problème en spécifiant l'ensemble des contraintes à respecter. Elles sont connues pour donner de bons résultats mais aussi pour leur temps de calcul extrêmement longs. Nous présenterons comment nous avons pu adapter une telle méthode pour un usage en temps réel. Pour finir, nous présenterons comment l'algorithme développé a pu être utilisé, avec un minimum de modifications, pour la planification de trajectoires et l'animation de mannequins articulés.

Enfin, nous conclurons ce mémoire en effectuant un bilan des travaux réalisés et en présentant les perspectives de recherche qu'ils permettent d'entrevoir.

Les développements présentés dans ce mémoire ont été réalisés en C++ et avec la bibliothèque WorldToolKit de Sense8 Corporation pour les parties visualisation et interaction 3D.

1 Historique

Le concept d'environnement virtuel n'est pas nouveau même si le terme n'a été introduit que récemment, sous la forme de *réalité artificielle*, par Myron Krueger [Krueger83]. Si l'on exclut les formes premières de créations comme la peinture, la sculpture, le théâtre ou le cinéma, c'est bien en 1956 que naquit la première expérience artificielle multisensorielle. C'est à cette date que Morton Heilig, jeune cinéaste inventif, créa le Sensorama qui permettait à un spectateur de s'asseoir au guidon d'une motocyclette et de parcourir les rues de Manhattan tout en ressentant odeurs, bruits (en stéréophonie), vibrations, vent et ce, devant des images en relief. Malheureusement, le Sensorama fut un échec commercial et fut rapidement oublié. Les photographies du Sensorama [Figure 2], publiées dans [Bormann94], montre cependant que, bien que multisensorielle, l'immersion n'était pas totale. En effet, l'utilisateur n'avait pas la possibilité d'influer sur le cours de la simulation ; il était limité à un rôle de spectateur.

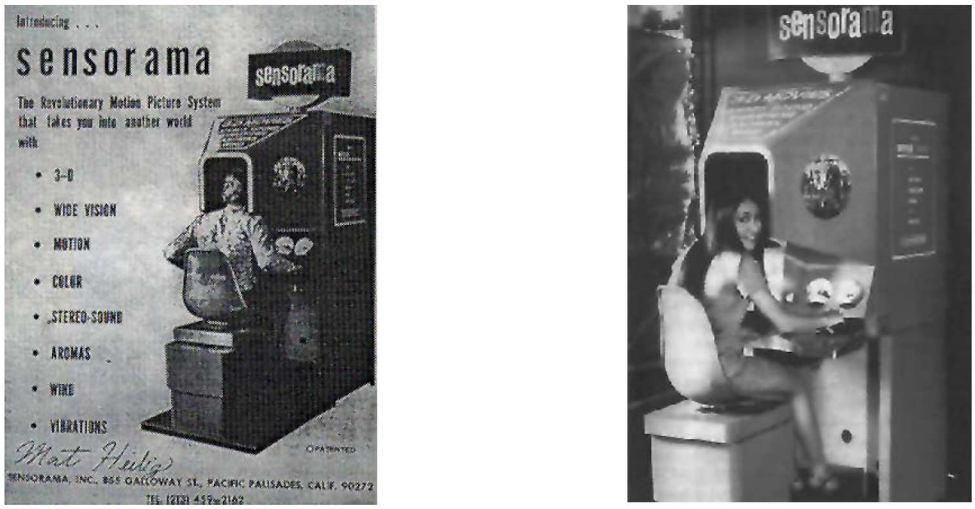


Figure 2 : *"Watch out for a remarkable new process called SENSORAMA! It attempts to engulf the viewer in the stimuli of reality. Viewing of the color stereo film is replete with binaural sound, colors, winds, and vibration. The original scene is recreated with remarkable fidelity. At this time, the system comes closer to duplicating reality than any other system we have seen!" Présentation du SENSORAMA, Morton Heilig, July 1964*

C'est quelques années plus tard, en 1961, qu'apparu la première solution permettant à un utilisateur de s'immerger dans un univers grâce à un casque de visualisation [Figure 3]. En effet, la société Philco Corporation proposait, pour la télésurveillance vidéo, un système de casque intégrant un mini-écran CRT (tube cathodique) ainsi qu'un capteur magnétique permettant de déterminer l'orientation de la tête de l'opérateur. Ainsi équipé, il percevait les images de la caméra de surveillance au travers de l'écran et pouvait piloter l'orientation de la caméra par ses mouvements de tête. Ce système, appelé Headsight, permettait à un utilisateur de surveiller des opérations dangereuses depuis un endroit sûr. Le concept de téléprésence était né sans que l'ordinateur n'ait été encore utilisé.

Figure 3 : *Le Headsight de Philco Corporation, 1961 - Image extraite de [Ellis95]*

Il est vrai que le travail sur ordinateur du début des années 60 se passait généralement en mode différé. Les utilisateurs soumettaient alors leurs programmes à l'opérateur qui avait pour charge de maintenir le planning d'utilisation de l'ordinateur et d'allouer les heures d'exécution pour chacun d'eux. Il était alors bien évidemment impensable d'imaginer pouvoir travailler de manière interactive avec un ordinateur. Il existait cependant une exception notable à cette règle. Le TX2, ordinateur à transistors construit par le Lincoln Laboratory du MIT, était utilisé uniquement en mode immédiat. Avec ses 320 Kilo-octets de mémoire, soit deux fois plus que la capacité maximale des ordinateurs du marché d'alors, le TX2 était de plus connecté à un moniteur et à un stylo optique [Figure 4]. En 1963, cette machine allait permettre à Ivan Sutherland de créer le premier programme graphique interactif, Sketchpad [Sutherland63]. Cette application [Figure 5], permettant le dessin de schémas scientifiques directement sur l'écran, possédait déjà la capacité de gérer des objets graphiques, de tracer des figures selon le principe de la boite élastique, etc. Il s'agit bien là de la première interface graphique, des années avant la création de ce terme.

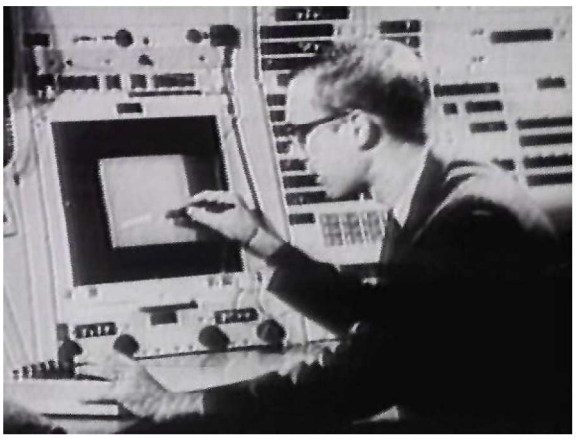


Figure 4 : *Ivan Sutherland devant la console du TX2*

Figure 5 : *Détail de la console du TX2*

En 1966, en tant qu'assistant professeur à Harvard, Ivan Sutherland et son étudiant, Bob Sproull, reprirent les expériences qu'avait menées la société Bell Helicopter sur le pilotage nocturne d'hélicoptères. Cette société avait elle-même poursuivi les travaux de Philco Corporation en associant un casque de visualisation à une caméra infrarouge, placée sur le nez de l'hélicoptère, afin de restituer aux yeux du pilote une réalité améliorée lui permettant d'atterrir de nuit.

Cependant, Ivan Sutherland voulait aller plus loin en remplaçant les images en provenance de la caméra par celles produites par un ordinateur. Il avait déjà, une année plus tôt, proposé l'idée d'immerger un utilisateur dans un monde simulé d'un réalisme tel qu'il le confondrait avec la réalité : « *The ultimate display* », écrivait Sutherland, « *would ... be a room within which the computer can control the existence of matter. A chair displayed in such a room would be good enough to sit in. Handcuffs displayed in such a room would be confining, and a bullet displayed in such a room would*

be fatal. With appropriate programming such a display could literally be the Wonderland in which Alice walked. » [Sutherland65]. Il reprit donc le système développé par Bell Helicopter, dans lequel il afficha une pièce tridimensionnelle en *fil de fer* [Figure 6] où l'utilisateur était immergé et se repérait grâce aux points cardinaux indiqués sur les murs. On était cependant encore bien loin du monde simulé parfaitement réaliste dont rêvait Ivan Sutherland mais le concept de réalité virtuelle tel qu'on l'entend aujourd'hui venait de voir le jour.

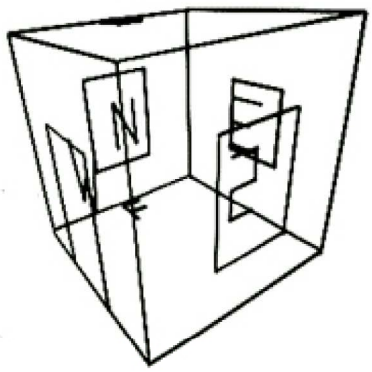


Figure 6 : *Alice au pays des merveilles selon Ivan Sutherland (1966)*

C'est en 1967, à l'université de Harvard, qu'Ivan Sutherland créa le premier casque de visualisation [Figure 7] affichant des images produites par un ordinateur. Il lui intégra même un système de localisation spatiale à base, dans un premier temps, de bras mécanique articulé [Figure 8] puis de détecteurs ultrasonores. Les informations de position et d'orientation en provenance du casque étaient alors utilisées par l'ordinateur pour générer des images en cohérence avec le point de vue de l'utilisateur.

Figure 7 : *Le casque de visualisation d'Ivan Sutherland*

Figure 8 : *Système de localisation associé*

Parallèlement aux recherches de Sutherland sur la visualisation interactive d'image numériques, Douglas Engelbart, créateur du SRI Augmentation Research Center en 1959, mena de nombreuses recherches sur les environnements de travail interactifs. Ces travaux se concentrèrent sur le système NLS (On Line System) qu'il mit au point au milieu des années 60 dans le but de développer et de

tester une nouvelle génération de logiciels et de périphériques qui devaient améliorer la productivité des utilisateurs de système informatiques. Les premières idées originales développées et implantées dans le NLS furent un système hypertexte ainsi qu'un système de téléconférence vidéo. En 1967, Douglas Engelbart publia [English67] les résultats de l'évaluation ergonomique de plusieurs types de périphériques pour l'interaction homme-machine. Plusieurs systèmes y étaient décrits parmi lesquels, un système étonnant permettant à l'utilisateur de contrôler un curseur à l'aide de son genou ainsi qu'un petit périphérique qui allait devenir universel : la souris [Figure 9] [Figure 10]. C'est d'ailleurs ce dernier appareil, créé en 1964, qui donnait et de loin les meilleurs résultats, s'avérant être alors le meilleur moyen pour un utilisateur de manipuler des objets à l'écran.

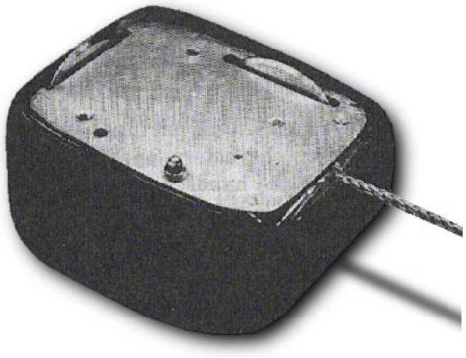


Figure 9 : *La souris de Douglas Engelbart vue de dessous*

Figure 10 : *Le même périphérique de coté*

Engelbart fut un des principaux pionniers de l'informatique interactive en inventant pour être intégré au NLS, entre autres, un éditeur de textes mêlant images et textes en deux dimensions, le concept de multifenêtrage ainsi qu'un système de commande d'ordinateurs à distance (Remote Procedure Call). Comme beaucoup de pionniers, Engelbart dut attendre plus de 15 ans pour voir ses travaux reconnus à leur juste valeur et intégrés dans les systèmes informatiques du commerce. Pour sa part, Fred Brooks de l'université de Nord Caroline, fut lui aussi un pionnier des techniques d'interaction entre l'homme et la machine. En 1967, il mit au point le système GROPE qui permettait à un opérateur de déplacer une particule autour d'une molécule virtuelle à l'aide d'un bras manipulateur. L'opérateur pouvait même ressentir, au travers du bras, les champs de forces simplifiés induits par la molécule sur la particule. Ce projet a, par la suite, connu plusieurs évolutions pour aboutir à un système, GROPE III, permettant la construction interactive de nouvelles molécules.

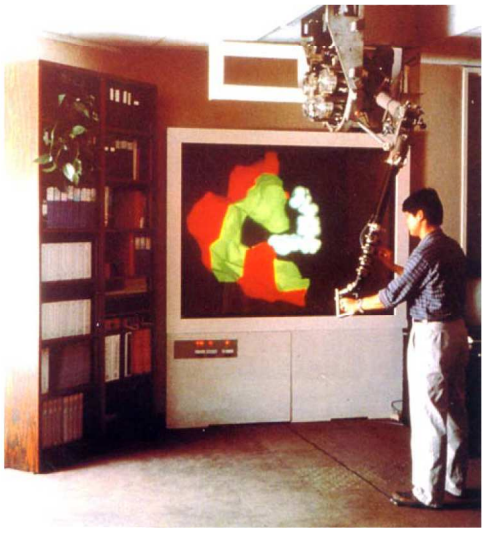


Figure 11 : *Le système GROPE III [Kalawsky93]*

C'est en 1977 que fut inventé le premier gant de données, le Sayre Glove, mis au point par Dan Sandin, Richar Sayre et Thomas Defanti de l'université de l'Illinois. L'objectif de ces chercheurs était alors de simplifier au maximum le dialogue entre l'homme et la machine afin de le rendre aussi transparent que possible. Pour cela, ils mirent au point un gant qui permettait de capter les mouvements de l'utilisateur, lui permettant ainsi de spécifier naturellement un ensemble de commandes gestuelles. Il fallut cependant attendre 1982 et l'invention de Thomas Zimmerman pour que soit définitivement ancré le terme de gant de données. Zimmerman eut l'idée d'utiliser des fibres optiques qui, placées sur chaque doigt de l'utilisateur, permettait d'en déterminer les mouvements. Ce dispositif permettait ainsi de proposer un périphérique [Figure 12] de faible encombrement et doté d'une bonne sensibilité. La rencontre entre Thomas Zimmerman, Jaron Lanier et Scott Fisher fut par la suite à l'origine de la création de la société VPL Research Inc qui allait cristalliser l'intérêt de la communauté scientifique et des médias sur les technologies de la Réalité Virtuelle.

Figure 12 : *Le Dataglove de VPL*

Parallèlement aux développements des périphériques d'interaction intuitifs, plusieurs dispositifs de visualisation furent mis au point. Là encore, le souci principal de leurs développeurs était de rendre l'utilisation de tels dispositifs aussi intuitive que possible. C'est ainsi qu'en 1984 McGreevy et Humphries, du NASA Ames Research Center, mirent au point un casque de visualisation stéréoscopique [Figure 13] à l'aide de deux écrans à cristaux liquides monochromes sur lesquels ils disposèrent des optiques afin d'accroître le champ de vue proposé. L'ordinateur utilisé générait alors une image différente pour chacun des deux écrans afin de restituer des images en relief.

Figure 13 : *Le casque de visualisation stéréoscopique de première génération de la NASA*

L'association de l'ordinateur, et de ses capacités à générer des images, aux périphériques pour l'interaction et la visualisation allait faire entrer l'informatique dans une ère nouvelle. L'homme, de spectateur, allait devenir acteur et enfin pouvoir influer sur le cours des expériences numériques qui lui seraient proposées.

2 Modélisation de scènes

La modélisation d'objets tridimensionnels par ordinateur est avant tout un processus de création visant à utiliser la machine comme un nouvel outil d'expression. Elle est utilisée pour retranscrire les *images mentales* d'un créateur sous une forme numérique qualifiée de *virtuelle*. Cette tâche se doit d'être la plus intuitive et la moins contraignante possible. En effet, elle ne doit imposer aucune contrainte liée à l'introduction de la machine dans le processus de création qu'elle a pour but, au contraire, de faciliter.

Depuis les premières années du développement de la synthèse d'images, les efforts de recherche se sont surtout portés sur les algorithmes de visualisation ([Watkins70], [Warnock69], etc.). Puis, depuis le milieu des années 70, alors que pratiquement aucune nouvelle technique de base ne voyait le jour, les efforts des chercheurs se sont portés sur les améliorations des techniques existantes (accélération des algorithmes, diminution de la mémoire requise, amélioration des modèles photométriques, etc.). C'est ainsi que la qualité des images obtenues n'a cessé de s'améliorer. Cependant, le niveau de performance des machines et des algorithmes, qui permettent aujourd'hui de visualiser de manière réaliste des scènes complexes, mettent en évidence une faille dans la chaîne de production d'une image. En effet, la création de scènes complexes reste encore et toujours un processus long et difficile qui n'offre, en général, que des opérations de trop bas niveau, imposant ainsi au créateur un effort d'adaptation au logiciel. Cet effort constitue en fait un réel frein à son activité créatrice.

Nous présentons, dans les paragraphes suivants, différentes représentations numériques des environnements virtuels ainsi que les méthodes et les solutions les plus probantes quant à la transcription d'une image mentale en une forme numérique tridimensionnelle.

2.1 Modélisation géométrique

Modéliser géométriquement un objet tridimensionnel ou plus généralement un environnement virtuel revient à définir l'ensemble des caractéristiques numériques qui caractérisent leur forme. Cet ensemble définit alors une représentation, ou modèle, qui peut être décrit de différentes manières [Sequeira 86].

2.1.1 La représentation par frontières (B-Rep)

Les objets sont construits à partir des éléments géométriques que sont les sommets, les segments et les polygones (ou facettes) [Figure 14]. Modéliser un objet tridimensionnel en définissant directement l'ensemble de ces éléments constitue une tâche extrêmement lourde. De plus, le modèle obtenu peut être très gourmand en mémoire ; il n'est pas rare qu'un objet complexe comporte plusieurs milliers de points et de facettes.

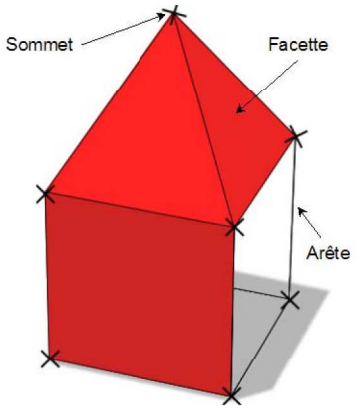


Figure 14 : *Description B-REP d'un objet 3D*

2.1.2 La construction géométrique de solides (CSG)

Les objets sont construits à l'aide de primitives de volumes élémentaires (cubes, sphères, cylindres, etc.), ou déjà composés, et d'opérations logiques de composition (union, différence, intersection, etc.). Un objet est donc défini par une arborescence [Figure 15] dans laquelle les nœuds de l'arbre représentent les opérations de composition et les feuilles, les primitives de volumes élémentaires.

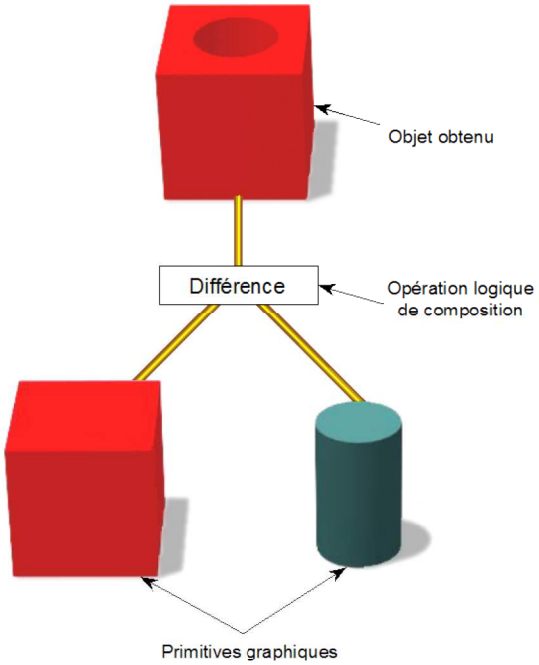


Figure 15 : *Description CSG d'un objet 3D*

Ce type de modélisation présente l'avantage d'être intuitif et d'offrir une possibilité de retour en arrière dans la construction d'un objet. On peut, en effet, retrouver sur la structure arborescente toutes les opérations effectuées, dans l'ordre chronologique, pour construire l'objet.

De plus, cette démarche est économe en mémoire puisqu'elle permet de définir un objet à partir de primitives.

2.1.3 La modélisation par éléments de volume

On considère que, dans ce cas, l'objet est contenu dans un cube divisé en cubes élémentaires [Figure 16]. Chaque cube élémentaire contient une information de base (densité, couleur, masse, etc.). Cette technique de modélisation est principalement utilisée dans les domaines médicaux où l'acquisition des données (scanner, RMN) constituant l'image se fait par tranches ou par coupes parallèles, leur réunion définissant ainsi l'objet considéré. Son principal inconvénient est la quantité importante d'informations à manipuler (qui peut être toutefois réduite en utilisant un découpage non plus régulier mais adaptatif par octree).

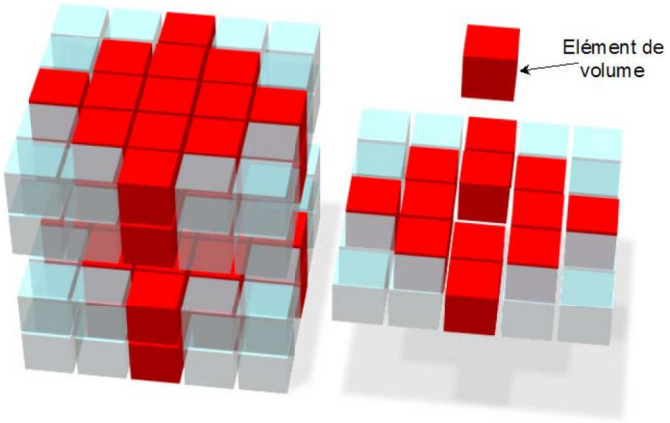


Figure 16 : *Modélisation volumique*

2.1.4 La modélisation par surfaces de forme libre

Les types de modélisation décrits précédemment permettent de construire des objets ayant une forme relativement géométrique. En revanche, ils ne permettent que difficilement de modéliser des formes complexes comme un visage, un fuselage ou une carrosserie. C'est pourquoi il est nécessaire de disposer d'outils géométriques permettant de définir de telles formes à l'aide de quelques points.

Les surfaces de forme libre [Figure 17] permettent ce genre de modélisation. Elles peuvent être surfaces de Bezier ou B-Splines et sont définies par des points de contrôle constituant un réseau (ou treillis) de contrôle. Elles possèdent en outre chacune leurs propres particularités [Sequeira 86].

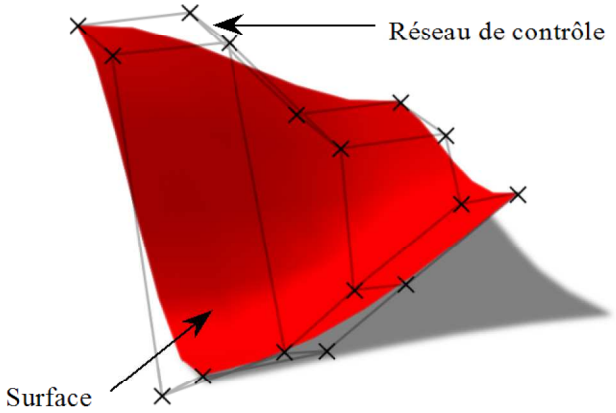


Figure 17 : *Surface de forme libre*

2.2 Modélisation géométrique par acquisition de données

Modéliser un objet par acquisition de données revient à utiliser un système de mesure permettant de repérer la position tridimensionnelle de points existants. Une fois obtenu le nuage de points définissant l'objet traité, une méthode de triangulation permet de transformer les données ponctuelles en données surfaciques ou maillage (ensemble de points 3D reliés entre eux par des arêtes formant ainsi un ensemble de faces).

Plusieurs méthodes d'acquisition de données 3D sont disponibles, des plus fastidieuses [Figure 18] aux plus perfectionnées, basées sur les principes de télémétrie laser ou de reconstruction quasi automatique à partir de séries de photos. Ces méthodes ne sont bien sur utilisables que pour modéliser des objets existants.

Figure 18 : *Des étudiants de l'université d'Utah mesurant la surface de la voiture d'Ivan Sutherland (photo de gauche) et le modèle obtenu, affiché par l'ordinateur (photo de droite)*

2.2.1.1 Acquisition par pointage

Cette méthode consiste à dessiner un maillage sur l'objet à numériser [Figure 18]. Il suffit ensuite de relever la position 3D de chacun des sommets en utilisant, par exemple, un stylet 3Draw® de Polhemus. Ce stylet contient un capteur magnétique miniature qui permet de localiser précisément une position dans l'espace.

La définition du maillage numérique sera ensuite réalisée à l'aide du logiciel d'acquisition, ou de tout autre outil de modélisation, en reliant chacun des points saisis par des arêtes définissant ainsi les facettes du modèle virtuel.

Cette méthode, bien que peu onéreuse, a l'inconvénient d'être fastidieuse et totalement inadaptée à la modélisation d'objets de grande taille (Par exemple, le volume utilisable avec le système de Polhemus est limité à 45*30*30 cm).

2.2.1.2 Télémesure laser

Cette méthode utilise les principes de triangulation laser plane [Figure 19]. Un faisceau laser balaie la scène à numériser pendant qu'un capteur CCD, parfaitement localisé par rapport à la source du faisceau, scrute le pic d'énergie produit sur l'objet touché.

Soit D_1 la droite directrice du rayon laser passant par sa source L et suivant une inclinaison, par rapport à l'horizontale, mesurée par un angle α connu du système. Cette droite est définie par :

$$M \in D_1 \Leftrightarrow \overrightarrow{LM} = k\vec{u} \text{ où } \vec{u} \text{ est le vecteur directeur de } D_1 \text{ défini par } \vec{u} = \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix} \text{ et } k \in \Re$$

Soit D_2 la droite passant par le centre optique C du capteur CCD et le point P', image du point P touché. Soit d la distance entre l'écran et le centre optique du capteur CCD, soit y l'ordonnée du point image P' obtenu, l'équation de la droite D_2 est alors de la forme :

$$M \in D_2 \Leftrightarrow \overrightarrow{CM} = k'\vec{v} \text{ où } \vec{v} \text{ est le vecteur directeur de } D_2 \text{ défini par } \vec{v} = \begin{pmatrix} d \\ y \end{pmatrix} \text{ et } k' \in \Re$$

Les coordonnées tridimensionnelles du point P sont alors calculées par l'électronique du capteur comme étant l'intersection de ces deux droites.

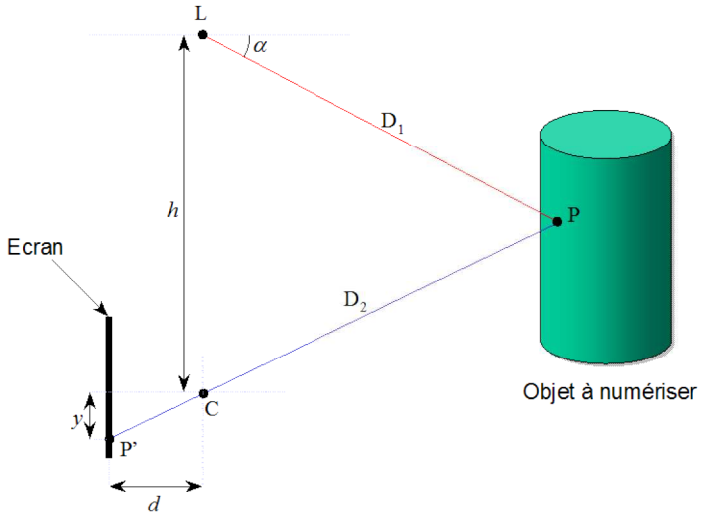


Figure 19 : *Principe de la triangulation laser plane*

L'enregistrement successif des positions obtenues fournit, au final, un nuage de points dense décrivant les surfaces traitées.

La méthode a pour avantages sa grande rapidité ainsi que la possibilité d'extraire, en association avec une caméra suivant le faisceau laser, les données photométriques (couleurs voire texture) des objets traités. De plus, la précision de la mesure, la possibilité de saisir une sous-zone de la scène en limitant l'intervalle de balayage et de choisir la densité du nuage généré en ajustant le pas de balayage constituent d'indéniables atouts. Son inconvénient réside dans le fait qu'elle fournit un nuage de points qui doit être ensuite traité manuellement par un opérateur afin d'en extraire les objets modélisés [Figure 22].

Le capteur SOISIC [Paramythioti93], conçu par la société MENSI avec le soutien d'EDF, fonctionne selon ce principe. La Figure 20 décrit une salle des machines (7m x 7m x 5m), traitées en 3 heures, et dont on génère un nuage de 250000 points [Figure 21]. Le modèle final est obtenu après traitement du nuage de points à l'aide du logiciel IPSOS3D [Thibault94]. Ce système propose, à l'utilisateur, un ensemble de primitives solides et de surfaces dont les dimensions, positions et orientations sont optimisées en fonction des points. EDF utilise cette solution pour la modélisation de centrales nucléaires ainsi que dans le cadre de plusieurs programmes culturels (Grotte Cosquer, site Grec antique de Delphes, Pont Neuf). Une caractéristique remarquable du capteur est sa capacité à numériser des scènes de très grande taille, dépassant les 100 mètres, avec une précision de l'ordre du millimètre à 5 mètres.

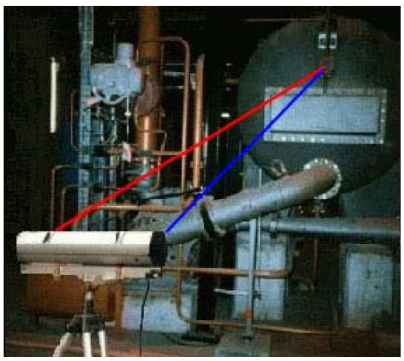


Figure 20 : *Le capteur SOISIC en environnement réel*

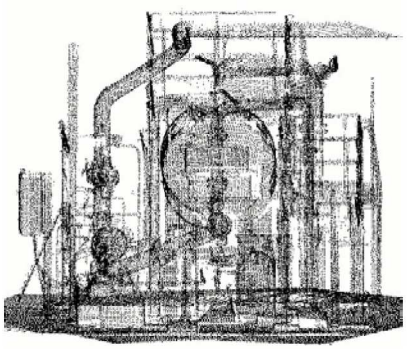


Figure 21 : *Nuage de points*

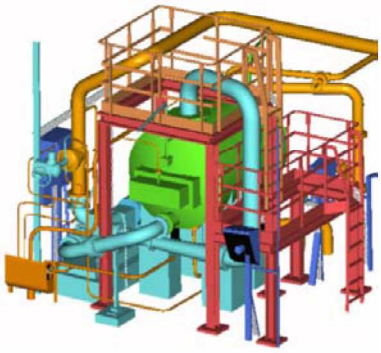


Figure 22 : *Modèle obtenu*

Les solutions [Figure 23] proposées par la société Américaine Cyberware sont basées sur les même principes, limités toutefois à la modélisation d'objets de petite taille. Contrairement au système SOISIC qui balaie la scène depuis un point fixe, les systèmes Cyberware tournent autour de l'objet à numériser ou, dans le cas d'objets de petite taille, font tourner ce dernier afin de modéliser la totalité de sa surface.

Les performances de ces systèmes leur permettent d'obtenir la position tridimensionnelle de quelques 5000 points par secondes avec une précision de l'ordre de 500 µm. Le nuage de points obtenu est triangulé de manière automatique (une facette triangulaire est construite pour trois points donnés) ; un algorithme de décimation de facettes est ensuite appliqué afin de simplifier le modèle 3D généré.

Figure 23 : *Les systèmes de numérisation 3D de Cyberware - Images Monolith Studios*

2.2.1.3 Télémesure optique

Il est aussi possible de faire une acquisition tridimensionnelle de positions à l'aide d'un système optique. Pour cela, on place un ensemble de marqueurs (balles blanches sur la Figure 24) sur l'objet ou le personnage à numériser. Ces capteurs sont ensuite repérés à l'aide d'un ensemble de caméras à haute fréquence de balayage (240 Hz) qui assurent un suivi fluide des mouvements.

Cependant, cette méthode n'est pratiquement utilisée que pour la capture de mouvements (« motion capture ») pour l'animation réaliste ou le suivi d'objets. Elle sera par exemple employée, dans le monde du jeu vidéo, pour acquérir le mouvement d'un joueur de football réel afin de définir l'animation réaliste de son double numérique.

D'autre part, elle pose notamment problème en cas d'occlusion des marqueurs, imposant l'intervention d'un opérateur humain pour déterminer les positions manquantes.

Figure 24 : *Système d'acquisition 3D optique avec marqueurs (à gauche) et caméra de suivi (à droite) - Images Monolith Studios*

2.2.1.4 Photogrammétrie

Cette méthode consiste à reconstruire un modèle tridimensionnel à partir d'un ensemble de photographies du modèle réel [Slama80]. Le principe de base consiste tout d'abord à numériser ces photographies afin de les fournir à la machine. Une fois cette opération effectuée, l'utilisateur doit fournir l'ensemble des caractéristiques qui définissent le système de prise de vue utilisé (longueur de focale, angle d'ouverture, zoom, paramètres de distorsion de l'objectif, etc.). Ce processus nécessite généralement une phase de calibration de la caméra afin de déterminer exactement ses paramètres de distorsion. L'utilisateur doit ensuite fournir au système la position et l'orientation précise de la caméra pour chaque photographie. Cette opération, généralement longue et délicate, peut être assistée par le système.

Une fois ces paramètres spécifiés, l'utilisateur détermine, pour chaque photographie, les points caractéristiques de l'objet photographié [Figure 25] [Figure 26].

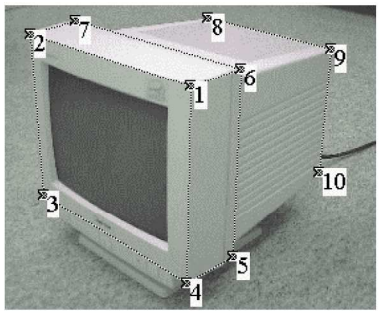


Figure 25 : *Points caractéristiques de l'image 1*

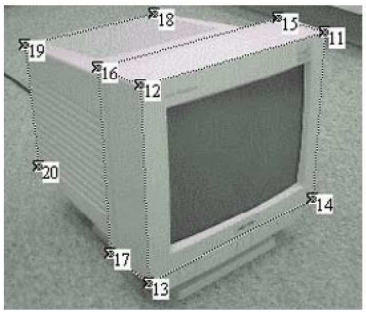


Figure 26 : *Points caractéristiques de l'image 2*

La phase suivante consiste à apparier manuellement ces points [Figure 27] [Figure 28], d'une photographie sur l'autre, le système retrouvant alors automatiquement les relations géométriques inter images.

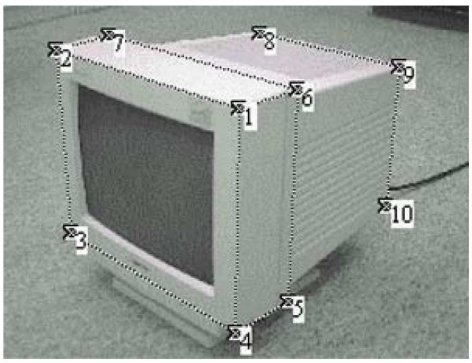


Figure 27 : *Points appariés de l'image 1*

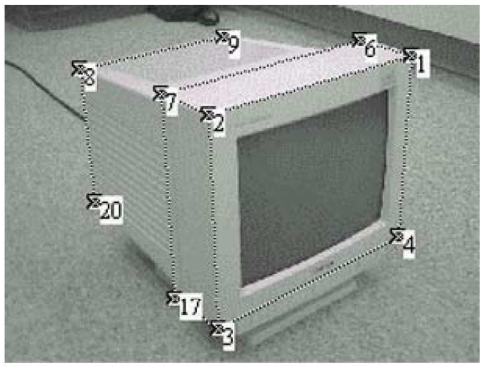


Figure 28 : *Points appariés de l'image 2*

L'utilisateur dispose, à ce stade du processus, d'un ensemble de sommets qu'il utilise ensuite pour définir le maillage de l'objet d'origine [Figure 29]. Les informations de texture sont généralement extraites automatiquement par le système utilisé.

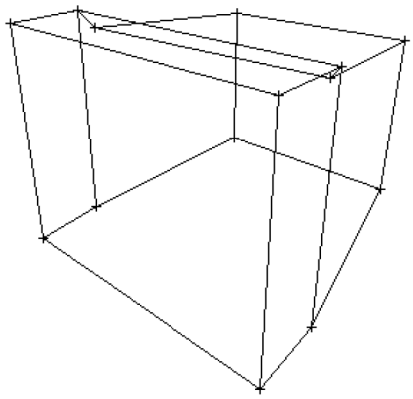


Figure 29 : *Modèle reconstruit par photogrammétrie*

2.2.1.5 Reconstruction à partir de séquence d'images

Le principe de cette méthode [Faugeras95] consiste à retrouver le modèle tridimensionnel d'une scène à partir d'une séquence d'images, sans information préalable sur les paramètres ou le mouvement de la caméra.

Pour cela, le système développé au sein du projet LTR REALISE de l'INRIA rétablit des correspondances [Luong92] entre les différentes images dont il retrouve la géométrie épipolaire [Zhang96]. C'est à partir de cette dernière que sont déterminées les matrices de projection correspondant à chacune des prises de vue. Cette opération permet de remonter à la structure de la scène à une transformation projective près.

Afin de déterminer la structure euclidienne, le système utilise une information a priori, qui porte soit sur la caméra, soit sur la scène. Si les paramètres de la caméra sont constants entre les différentes vues, une méthode automatique peut être appliquée, sinon, des informations sur la scène doivent être fourni, à priori, afin de déterminer la transformation projective manquante. En pratique, ces informations sont spécifiée par l'utilisateur. En utilisant les relations de parallélisme entre droites visibles dans les images, le système détermine la structure affine de la scène. Enfin, la connaissance de certains angles ou de certains rapports de longueurs permet de remonter à la structure euclidienne.

Une fois les vues calibrées, il reste à construire le modèle géométrique de la scène. L'approche adoptée est semi-automatique. L'utilisateur définit un modèle polyédrique en interagissant directement avec les images. Il reçoit pour cela l'assistance du système qui permet l'extraction précise de coins ou de jonctions ainsi que la mise en correspondance automatique de primitives entre les images. La précision du maillage est par conséquent laissée au choix de l'opérateur. Les textures destinées à être plaquées sur les polygones sont extraites automatiquement des images. À l'aide des techniques de calcul de mosaïques, les informations de texture extraites des différentes images sont combinées, permettant ainsi l'élimination automatique d'une grande partie des artefacts visuels (piétons, voitures) ainsi que le lissage photométrique des textures (différence de teinte ou de luminosité).

La Figure 30 présente deux exemples de modèles obtenus à l'aide de ce système. À gauche, nous montrons une des images utilisées, et le modèle filaire reconstruit. À droite, nous montrons deux vues du modèle texturé produit.

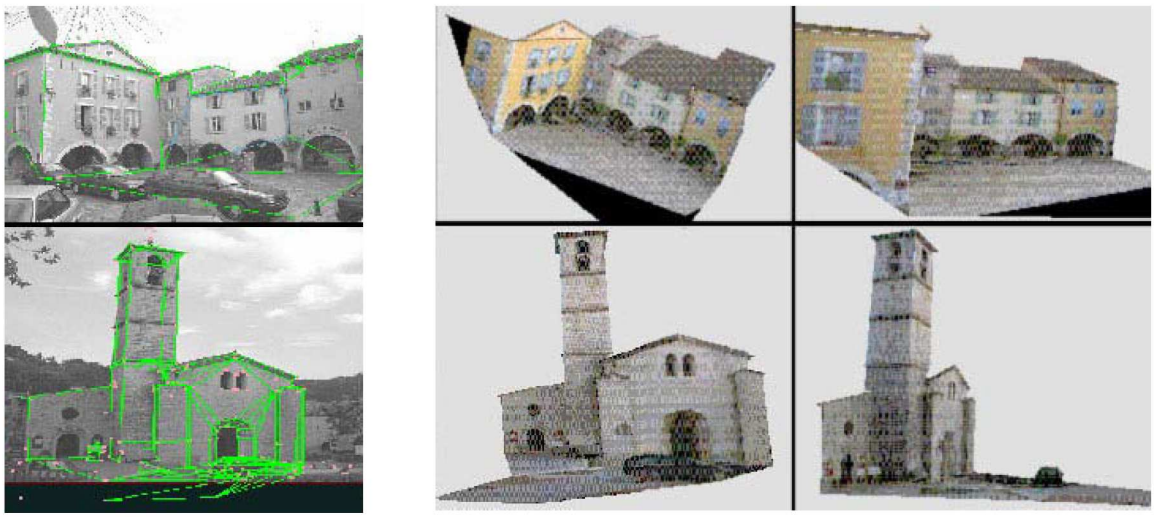


Figure 30 : *Reconstruction d'un modèle numérique (photos de droite) à partir de deux prises de vue différentes (photos de gauche)*

2.3 La modélisation de scène

Une scène est avant tout un ensemble d'objets auxquels on attribue des informations de position et d'orientation. On y rajoute habituellement l'ensemble des lumières qui constituent l'éclairage ainsi qu'un certain nombre de caméras virtuelles définies par leur position, orientation et leurs propriétés optiques.

Cette définition peut être étendue par l'ajout de comportements ou de propriétés plus spécifiques en fonction du format de description choisi. En effet, ce dernier offre plus ou moins de possibilités à l'utilisateur pour définir les propriétés de la scène. Les formats disponibles sont multiples et variés et peuvent être limités à une simple description géométrique (DXF) ou permettre la définition de comportements complexes (VRML97) voire de propriétés mécaniques (CATIA, EUCLID).

Malgré cette diversité, les données composant la scène sont pratiquement toujours organisée sous une forme hiérarchique décrivant les notions d'attachement et de composition d'objets.

2.3.1 Le graphe de scène

Le graphe de scène [Figure 31] est la forme de description hiérarchique la plus communément utilisée. On appelle nœud l'élément de base de ce type de description qui peut, ou contenir différents types de données (géométrie, lumière, position, atmosphère, etc.), ou être lui même sous-graphe de scène.

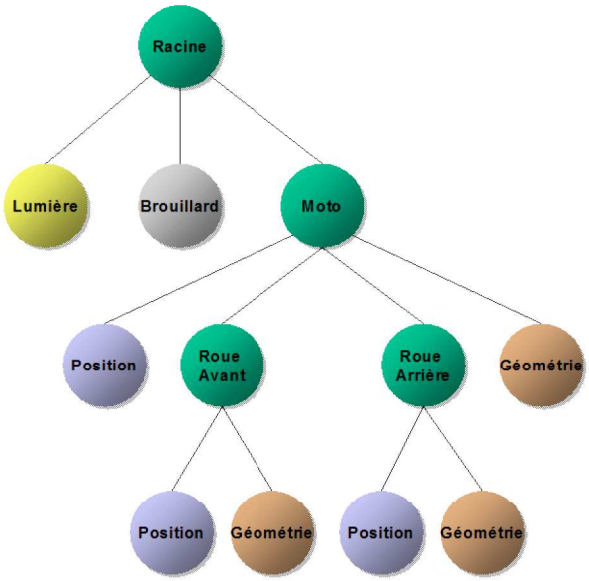


Figure 31 : *Un graphe de scène simplifié*

2.3.1.1 Terminologie

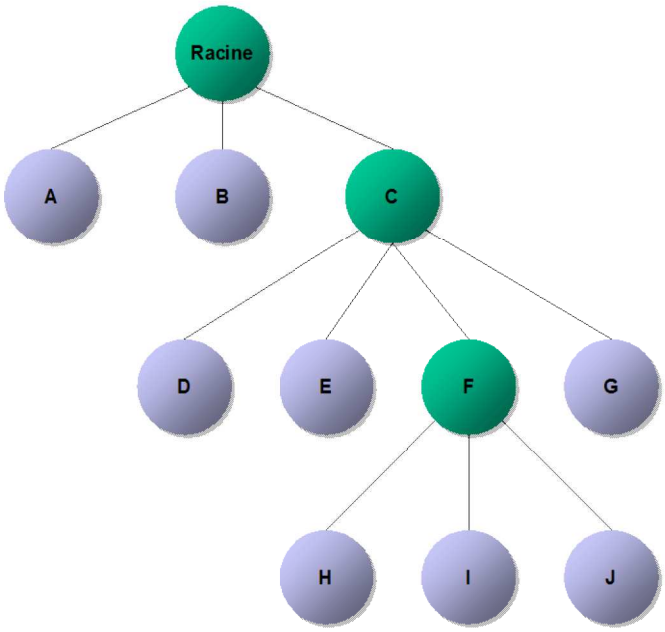


Figure 32 : *Vue schématique d'un graphe de scène*

Ancêtre : si le nœud C est un sous-arbre contenant un nœud H alors C est dit ancêtre de H.

Descendant : tout nœud contenu dans le sous-arbre d'un autre nœud est dit descendant de ce dernier.

Enfant ou Fils : tout nœud étant rattaché directement à un autre est dit enfant, ou fils, de ce dernier. Dans la Figure 32, F est fils de C tout comme E est enfant de C.

Feuilles : une feuille est un nœud n'ayant pas d'enfants. Les nœuds A, B, D, E, H, I, J et G sont les feuilles du graphe présenté Figure 32.

Ordre de traitement : l'ordre de traitement du graphe est l'ordre dans lequel sont traités l'ensemble des nœuds.

Parent ou Père : l'ancêtre direct d'un nœud est appelé parent, ou père, de ce dernier. Dans la Figure 32, C est père de E.

Prédécesseur : si le nœud A est traité avant le nœud B alors A est dit prédécesseur de B.

Racine : il existe un et un seul nœud racine dans tout le graphe de scène. Il est l'ancêtre de tous les autres nœuds.

Sous-arbre : un nœud possédant des enfants est dit sous-arbre. Les nœuds C et F sont les sous-arbres du graphe présenté Figure 32.

2.3.1.2 Conclusion

L'ensemble des nœuds est organisé sous la forme d'un graphe acyclique direct. Ce type de structure offre des possibilités de description de scènes très puissantes tout en restant parfaitement adapté aux simulations 3D temps réel. Il permet, entre autres, de prendre en compte les propriétés de découpage fonctionnel et spatial de la scène afin d'en faciliter ses manipulations ainsi que sa vitesse de traitement. L'utilisateur a par exemple la possibilité de regrouper les objets, proches dans l'espace, sous un même nœud groupe. Cette fonctionnalité permet d'améliorer l'algorithme de rejet (« culling ») qui consiste à ignorer les nœuds qui ne sont pas visibles. En effet, plutôt que de traiter un à un tous les nœuds d'un même groupe afin d'étudier leur visibilité, l'algorithme de rendu sera capable de rejeter tous ces nœuds, s'ils sont invisibles, en ne traitant que le nœud groupe. De même, pour déplacer un ensemble de nœuds, il est bien plus efficace d'appliquer une translation au nœud père qu'appliquer cette même translation à chacun de ses fils.

2.3.2 Le langage VRML

Le langage VRML (Virtual Reality Modeling Language) [Ames96] a été créé pour la description de simulations interactives multi-utilisateurs. Ces simulations sont définies par des environnements virtuels interconnectés au travers du réseau Internet et pouvant faire référence à des documents du World Wide Web à l'aide de liens hypermédias. Tous les aspects de ces simulations interactives devaient pouvoir être définis, à savoir la programmation de comportements, de schémas d'interaction aussi bien que les procédures de communication réseau.

En réalité, VRML, dans sa version première, est resté limité à la modélisation de scènes.

2.3.2.1 Historique

C'est en 1994 à Genève, lors de la première conférence sur le World Wide Web, que Tim Berners-Lee et Dave Raggett organisèrent une session pour débattre des connections possibles entre les systèmes de réalité virtuelle et le Web.

Lors de cette session, plusieurs projets en court furent présentés ; tous visaient à concevoir un système de visualisation graphique 3D pouvant interopérer avec le Web. Il devenait alors évident que ces projets devaient pouvoir bénéficier d'un langage commun pour la description de scènes 3D et de liens hypermédias, un langage équivalent à HTML (HyperText Markup Language) utilisé pour définir les pages consultables sur le Web. Le terme VRML (Virtual Reality Markup Language) était créé, la phase de conception pouvait alors débuter. La signification de l'acronyme VRML fut par la suite changée en « Virtual Reality Modeling Language » afin de mieux mettre en valeur la nature graphique de ce langage.

L'originalité de la genèse de VRML réside dans le grand nombre de personnes qui y furent impliquées, dès la phase de spécification. En effet, le choix fut fait d'utiliser le puissant outil de communication que représente Internet, en mettant en place la liste de diffusion www-vrml où plus d'un millier de personnes échangèrent idées et points de vue. Une date fut fixée pour la livraison d'une version draft du document de spécification. Cette date, fixée pour l'automne 1994 au cours de la conférence WWW'94, ne laissait que peu de temps pour achever le travail en cours. C'est pourquoi il fut décidé d'adapter la première version de VRML à partir d'une solution existante : le format Open Inventor de Silicon Graphics Inc. allait être retenu et modifié afin de supporter les extensions réseau.

2.3.2.2 Identification des besoins

Le document final de spécification de la version 1 [Bell95], produit en mai 1995, identifie trois besoins à satisfaire pour la première version du langage :

- indépendance vis-à-vis de la plate-forme informatique choisie,
- extensibilité,
- possibilité de fonctionner efficacement sur des réseaux à bas débit.

Il fut aussi décidé de ne pas concevoir VRML comme une extension du langage HTML. En effet, HTML a été créé pour la description de documents textuels ; de part la quantité de données mise en œuvre et sa contrainte temps réel, la visualisation de scènes 3D disponibles sur le réseau requiert de plus grandes optimisations que celle de simple documents textuels. De plus, HTML était déjà, à l'époque de la conception de VRML, un standard accepté et très répandu. Concevoir VRML comme une extension d'HTML aurait risqué d'imposer des contraintes de compatibilité très limitatives.

Il fut aussi décidé de ne pas offrir de capacité descriptive de comportements complexes ou interactifs dans la première version de VRML. En effet, de telles fonctionnalités requièrent la mise à disposition d'un langage bien plus complexe que celui nécessaire à la description géométrique de scènes. Certes, plusieurs langages existaient alors mais en choisir un aurait risqué de déclencher des conflits de choix technologiques entre les différentes personnes impliquées dans le projet VRML. La définition d'un langage pour la programmation de comportements fut donc reportée à une version ultérieure de VRML.

2.3.2.3 Spécifications du langage

D'un point de vue général, VRML représente un moyen pour lire et écrire des objets 3D pouvant théoriquement contenir de nombreux types d'informations (données géométriques, sons, images, etc.).

Le choix du type de structure de données s'est naturellement porté sur une structure hiérarchique de type « graphe de scène » intégrant la notion de propagation des propriétés ; les nœuds traités en premier ont une influence sur les suivants, quelque soit leur position dans le graphe. On définit la notion d'état du graphe par les valeurs d'un ensemble d'attributs utilisés pour l'affichage, à savoir la valeur de l'éclairage, de matériau, la matrice de transformation ainsi que les données atmosphériques.

La valeur de ces variables est modifiée au fil du traitement du graphe ; ce parcours est du type « en profondeur d'abord » [Figure 33]. Dans le cas de transformations géométriques (translation, rotation, échelle), on parle d'accumulation de transformations. En effet, la valeur de transformation courante n'est pas simplement remplacée lors de la rencontre d'un nouveau nœud transformation. Au contraire, on lui multiplie la valeur de transformation du nœud traité afin d'obtenir la nouvelle valeur courante.

Encore critiqué de nos jours, le choix du type de propagation des propriétés découle de l'utilisation d'Open Inventor comme base de travail. Un mécanisme est cependant disponible pour limiter les effets de la propagation des propriétés d'un nœud ; un nœud séparateur [Figure 34] pourra être utilisé pour isoler fonctionnellement une partie du graphe des autres parties.

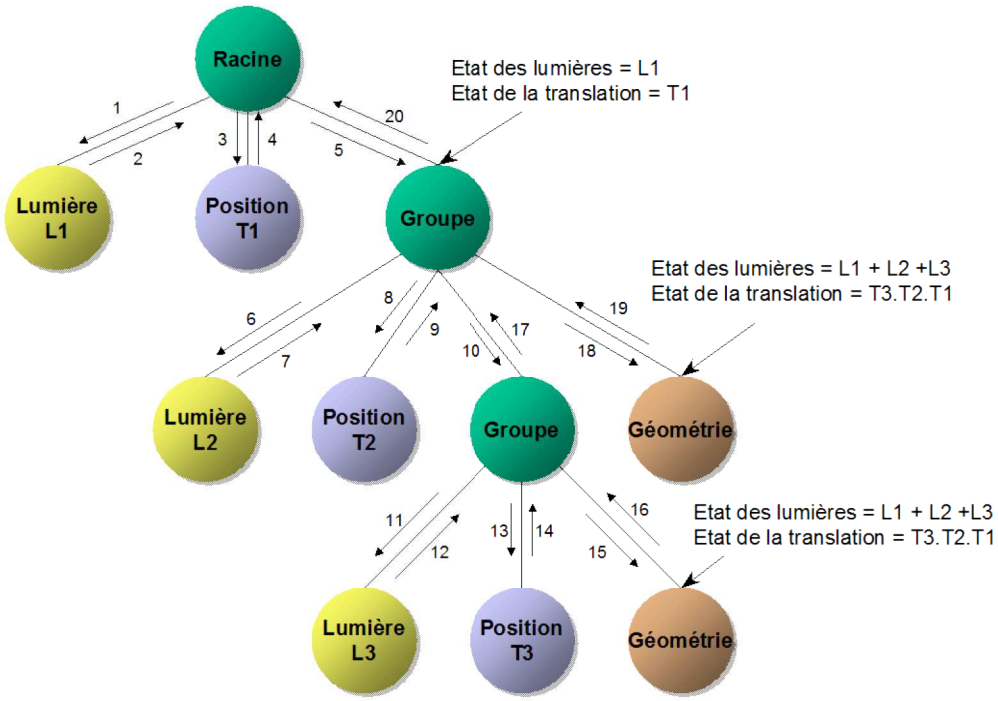


Figure 33 : *Ordre de parcours du graphe et modification de son état*

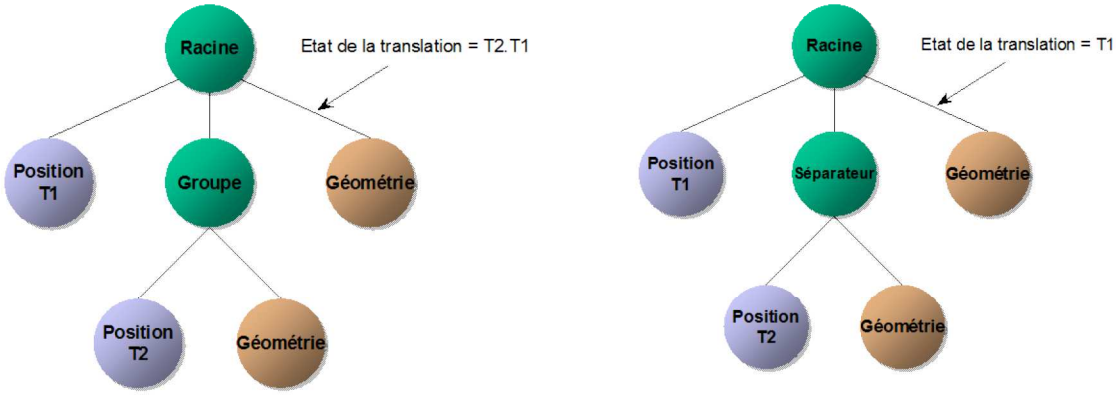


Figure 34 : *Limitation de la propagation des propriétés d'un nœud par insertion d'un séparateur*

La norme VRML définit plusieurs types de nœuds possédant tous les attributs suivants :

- le type du nœud,
- le nom du nœud,
- les paramètres qui différencient celui-ci des autres nœuds du même type (par exemple, le rayon pour un nœud de type sphère),
- la liste des nœuds fils dans le cas où le type du nœud lui permettrait de contenir des enfants.

2.3.2.4 Principaux types de nœuds

Il existe 4 grandes classes de types de nœuds :

Les nœuds pouvant regrouper des enfants :

Group : un nœud de type Group peut contenir de 0 à n enfants. Il permet de regrouper un ensemble de nœuds sous une même racine.

Separator : les nœuds de ce type ont la même fonction que ceux du type Group à ceci près que les propriétés des enfants du nœud Separator ne sont pas propagés à ses frères [Figure 34].

Switch : un nœud de type Switch regroupe lui aussi plusieurs nœuds. Cependant, il offre la possibilité de n'activer aucun, un seul ou tous ses fils ; un nœud désactivé sera alors ignoré lors du traitement du graphe de scène.

LOD : un nœud de type LOD (Level Of Detail) permet de regrouper plusieurs nœuds auxquels sont associés des intervalles de visibilité. Un seul de ces nœuds est activé à la fois. Celui-ci est déterminé en fonction de la distance du nœud LOD au point de vue et de son intervalle de visibilité.

Les nœuds définissant géométriquement une forme :

Les nœuds de type **AsciiText**, **Cone**, **Cube**, **Cylinder** et **Sphere** représentent les primitives graphiques disponibles avec VRML.

Il est aussi possible de définir des objets géométriques complexes à l'aide de nœuds de type liste de faces (**IndexedFaceSet**), de lignes (**IndexedLineSet**) ou de points (**PointSet**).

Les nœuds définissant des propriétés d'affichage et d'éclairage :

Il existe différents types de nœuds permettant la définition des sources d'éclairage de la scène (**PointLight**, **SpotLight**, **DirectionnalLight**).

De plus, plusieurs type de caméras sont disponibles (**PerspectiveCamera**, **OrthographicCamera**).

Les nœuds faisant référence à des fichiers externes :

WWWInline : un nœud de type WWWInline fait référence à un fichier VRML accessible sur le World Wide Web, le nom de ce fichier étant spécifié sous la forme d'un URL (Uniform Resource Locator [URL95]). Lors du chargement du graphe de scène, le nœud WWWInline sera remplacé par le fichier auquel il fait référence, à condittion bien sur qu'il soit au bon format.

WWWAnchor : un nœud de type WWWAnchor fait lui aussi référence à un fichier externe au travers d'un URL. Contrairement au cas précédent, le fichier référencé peut être d'un type quelconque. Il sera donc fait appel à un autre moyen de restitution dans le cas où ce fichier serait à un format autre que VRML (son, fichier MIDI, film, etc.).

2.3.2.5 Instances de nœuds

Comme nous l'avons précisé dans les chapitres précédants, la description géométrique d'objets complexes peut être grande consommatrice de mémoire. C'est pourquoi la norme VRML propose un mécanisme d'instanciation afin d'économiser la mémoire dans le cas où un même nœud serait utilisé à plusieurs endroits dans le graphe : Au lieu de dupliquer les données d'un nœud, il est possible de ne les stocker qu'une seule fois et d'y faire référence au travers d'un nœud instance [Figure 35]. Par exemple, la description géométrique d'une roue de voiture ne sera faite qu'une seule fois, les trois autres roues seront ensuite définies par instanciation de la première.

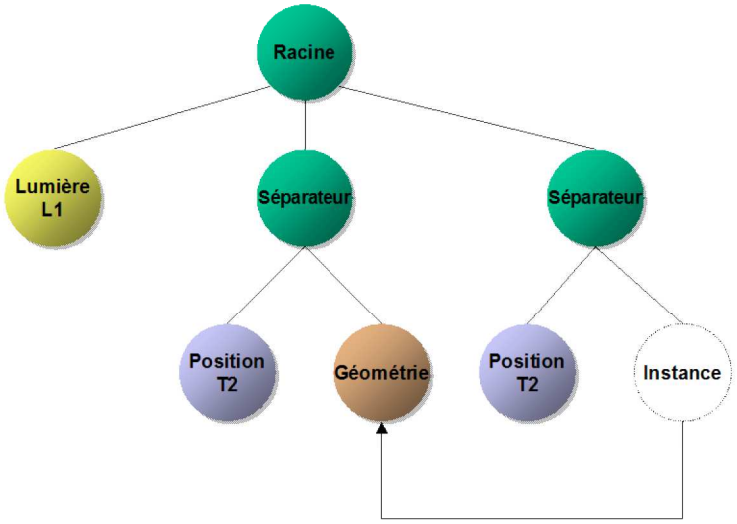


Figure 35 : *Instance de nœud géométrique*

2.3.2.6 Extensibilité

Cette fonctionnalité est assurée par la possibilité qu'à l'utilisateur de définir de nouveaux types de nœuds. Pour cela, il doit décrire le nouveau type en fournissant tout d'abord la liste des attributs qu'il utilise. Cette étape revient à fournir les spécifications du nouveau type.

Une définition du type Cube pourrait être, par exemple :

```
Cube {
fields [SFFloat width, SFFloat height, SFFloat depth]
width .1 height .5 depth .3 …
```

Un mécanisme d'héritage simple est de plus disponible pour créer de nouveaux types à partir de ceux déjà existants. Ce mécanisme est cependant rarement pris en compte par les outils compatibles VRML.

2.3.2.7 Conclusion

Certes, le format VRML est devenu l'un des standards de description de scènes les plus utilisés actuellement mais il aura fallu attendre plusieurs années avant que les outils à la disposition des programmeurs ne soient stabilisés et puissent charger ou sauver correctement les fichiers respectant ce format. Il est d'ailleurs facile de se convaincre des difficultés qu'ont eues les développeurs à fournir des fonctions d'import ou d'export VRML correctes. Pour cela, il suffit de prendre une scène, un minimum complexe, pour s'apercevoir qu'en 1997 encore, chaque logiciel compatible VRML proposait sa propre interprétation du standard, à savoir un affichage plus ou moins exotique.

Les choses s'étant suffisamment stabilisées en 1997, VRML peut désormais être raisonnablement considéré comme un standard réel et efficace pour la description de scènes. Cependant, nos observations sont suffisamment étayées et importantes pour que l'on puisse légitimement émettre des réserves quant à l'utilisation industrielle, dans un avenir proche, de la nouvelle version du langage, VRML97, bien plus complexe que la précédante.

L'impossibilité de l'utiliser ce format sous forme binaire, et donc compacte, reste toutefois un de ses plus gros désavantages ; un fichier VRML ne peut être écrit qu'en ASCII, atteignant ainsi des tailles mémoire énormes lorsqu'il contient la définition géométrique des objets de la scène. Toutefois, pour des soucis de compatibilité et de portabilité, nous avons choisi d'utiliser ce langage dans tous les travaux présentés dans ce mémoire.

2.4 Modélisation géométrique et temps réel

Bien qu'extrêmement lourd et par là même peu intuitif, le modèle B-Rep n'en reste pas moins la description géométrique qui doit être fournie au calculateur graphique afin qu'il puisse afficher l'objet modélisé en temps réel. En effet, la simplicité de traitement de ce modèle a permis d'implanter, au sein de microprocesseurs dédiés, les algorithmes optimisés pour le traitement de points 3D et de facettes. La très grande majorité des cartes graphiques 3D du commerce sont en fait de vrais pipelines dans lesquels on injecte les points et les facettes et qui, au final, produisent des images.

L'inadéquation entre les données requises par l'ordinateur et celles qu'un humain est à même de manipuler, naturellement, demeure encore un réel paradoxe. C'est pourquoi plusieurs systèmes de modélisation, plus ou moins évolués, ont vu le jour tout au long de l'histoire de la conception assistée par ordinateur (CAO). Leur objectif premier est d'offrir à l'utilisateur des outils de haut niveau permettant de définir, à moindre effort, les caractéristiques géométriques des objets tout en faisant abstraction du modèle de représentation sous-jacent.

Pour cela plusieurs techniques ont été développées afin de proposer des systèmes de modélisation puissants et conviviaux qui, il est important de le noter, restent la plupart du temps réservés à des spécialistes [Figure 36].

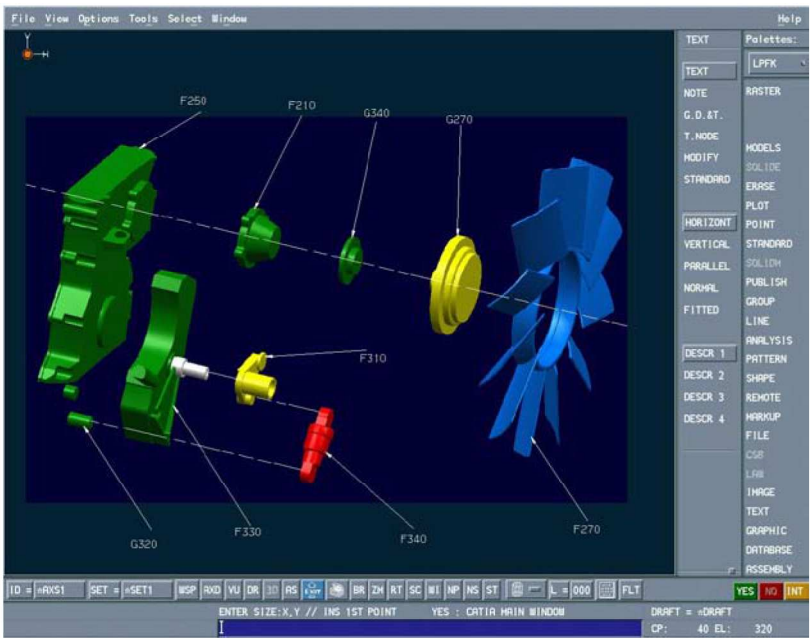


Figure 36 : *L'interface graphique du logiciel CATIA de Dassault Systèmes*

2.5 La modélisation géométrique par contraintes

De nombreux systèmes de conception assistés par ordinateur (CAO) sont basés sur le modèle d'un ensemble d'objets liés entre eux par des contraintes. Ce modèle impose que les contraintes soient respectées quelques soient les opérations effectuées par l'opérateur. Ces systèmes utilisent, pour cela, un ensemble de relations entre différentes variables, définissant ainsi les contraintes (distance, angle, tangence, incidence, etc.) appliquées aux éléments du modèle (points, droites, cercles, etc.). Le changement d'état d'une ou plusieurs de ces variables entraîne alors les modifications nécessaires pour que toutes les relations soient à nouveau satisfaites. De tels systèmes peuvent être classés en deux catégories selon l'approche qu'ils utilisent.

2.5.1 Approche numérique

Les systèmes utilisant ce type d'approche transforment les contraintes géométriques en un système d'équations, linéaires ou non, qui est résolu à l'aide de méthodes numériques.

L'introduction, dans le monde de l'infographie, du concept de contraintes remonte aux année 60 avec l'apparition de Sketchpad [Sutherland63]. Ce système permettait d'imposer, aux éléments graphiques, des contraintes géométriques exprimées sous formes d'équations. L'ensemble de ces équations était représenté sous la forme d'un graphe dont on étiquetait les arêtes par les valeurs numériques des cotes des éléments traités. Le système formé était alors résolu [Sutherland63] par une méthode de propagation des degrés de liberté ou, si elle échouait, de relaxation, assurant ainsi la satisfaction d'un ensemble maximal de contraintes.

C'est vers la fin des année 70 que A. Borning développa Thinglab [Borning81], logiciel de simulation fournissant un environnement pour construire des modèles dynamiques d'expérimentation en géométrie, physique, etc. Thinglab intègre un solveur de contraintes implanté avec le langage à objets Smalltalk. Il manipule, en outre, des objets qui communiquent entre eux par envoi de messages. Chaque classe d'objet est définie par les classes dont elle hérite, la description des parties qui la composent et la liste des contraintes qu'elle doit satisfaire. Chaque contrainte est définie par une méthode permettant de savoir si elle est satisfaite et un ensemble de méthode utilisées pour la satisfaire.

Un système implanté selon le paradigme objet a été décrit par C. Nguyen [Nguyen92] afin de concevoir et de manipuler formellement des solides (construits suivant le modèle CSG) paramétrés par l'utilisation de contraintes. Chaque solide est défini par son repère local, sa dimension, sa position, sa forme et ses contraintes. Suivant le type du solide, plusieurs paramètres sont utilisés pour définir sa dimension, comme le rayon pour une sphère ou le rayon et la hauteur pour un cylindre.

Plusieurs types de contraintes géométriques ou topologiques sont utilisables. Les contraintes géométriques sont du type, par exemple, « droite D1 distante de la droite D2 », « point P centre de la face F », etc. De même, les contraintes topologiques peuvent être « solide fixe » ou « solide à n faces ». La résolution des contraintes, confiée à un système de calcul formel, est algébrique (il est cependant possible d'utiliser une méthode de résolution numérique du type Newton-Raphson).

L'approche numérique permet de traiter de nombreux types de contraintes. Cependant, les techniques utilisées ne convergent pas toujours vers une solution. De plus, le nombre d'équations qui composent le système est directement proportionnel à la complexité de la scène, imposant ainsi l'utilisation d'algorithmes de décomposition de systèmes d'équations.

2.5.2 Approche déductive

Le principe de cette approche consiste à utiliser un moteur d'inférence pour résoudre un problème (i.e. un ensemble de contraintes à satisfaire) défini par un ensemble de règles.

Par exemple, B. Brüderlin [Brüderlin86] utilise le langage Prolog pour résoudre des problèmes géométriques dans un espace à deux dimensions. Son système manipule des points, des segments ou des cercles auxquels il associe des contraintes au moyen de règles Prolog. Le calcul d'une solution pour un problème donné se déroule en deux étapes. La première s'effectue à un niveau symbolique et consiste en l'élaboration d'un ensemble de règles de construction de la figure désirée. Cette étape permet de détecter et d'analyser les problèmes d'incohérence dans la description. Enfin, c'est lors de la deuxième étape que les procédures d'inférence de Prolog évaluent numériquement la solution.

S. Ait Aoudia présente, dans [AitAoudia92], un système qui permet de déduire (définir) une entité géométrique des contraintes qui lui sont appliquées. Ce système est conçu à l'aide d'un formalisme conforme au paradigme objet (Smeci). Les objets élémentaires sont le point, la droite et le plan. Les contraintes géométriques permettent de spécifier des propriétés géométriques sur les constituants d'une scène (angle entre deux plans, distance d'un point à un autre, etc.). Elles sont définies à l'aide de règles écrites en LISP et fournissent les informations nécessaires pour fixer un ou plusieurs degrés de liberté sur un objet. Un objet est dit complètement déterminé lorsqu'il n'a plus de degré de liberté. Le système construit, dans cet objectif, un arbre de contraintes dont les sommets, représentant les entités géométriques, sont reliés entre eux par des arcs représentant les contraintes. Lorsqu'une contrainte est modifiée (changement d'angle, etc.), le système exécute une procédure pour vérifier si l'ensemble des contraintes est toujours satisfait et, le cas échéant, réajuste les entités affectées par la modification.

Bien que permettant la programmation élégante d'un ensemble de contraintes, la résolution déductive reste lourde et grosse consommatrice de temps machine et de mémoire. Elle demeure, aujourd'hui encore, inappropriée pour les systèmes soumis à de fortes contraintes temps réel .

2.6 La modélisation déclarative de scènes

Les systèmes présentés ci-dessus souffrent tous du manque d'abstraction qu'ils proposent à l'utilisateur. En effet, ils restent trop proches du modèle géométrique sous-jacent et créent ainsi une barrière qui sépare ce que veux faire l'utilisateur de ce qu'il arrive à faire à l'aide des fonctionnalités offertes. De plus, la création d'une forme ou d'une scène nécessite, avec ces systèmes, la définition de tous ses composants, opération devenant rapidement fastidieuse et contraignante. C'est pourquoi plusieurs études ont été menées dans le but d'offrir, à l'utilisateur, les outils qui lui permettraient de retranscrire, le plus naturellement possible et sous une forme numérique, l'image qu'il a d'une scène ou d'une forme. L'approche déclarative est caractérisée par le haut niveau d'abstraction des opérations qu'elle propose à l'utilisateur, le libérant ainsi de toute considération numérique. Par exemple, l'utilisateur peut disposer de commandes du type « poser_sur » ou « poser_à_coter_de ». Cependant, de telles commandes ne peuvent être aussi précises, géométriquement parlant, que les commandes mathématiques proposées dans les systèmes de modélisation traditionnels. Ainsi, comment peut on déterminer de manière précise la configuration à laquelle pense l'utilisateur lorsqu'il exprime une commande du type « pose le verre sur la table » ?

Afin de palier ce manque de précision, deux stratégies sont envisageables. La première consiste à proposer l'ensemble complet des solutions possibles. Cependant, son cardinal peut être très élevé, posant ainsi le problème de l'exploration de l'univers des solutions afin d'en choisir une. Cette approche, proposée dans le projet MultiFormes [Plemenos91], semble donc inappropriée pour un utilisateur qui aurait une idée très précise à réaliser.

2.6.1 Les projets ExploFormes

Parmi ces études, le projet ExploFormes, initié par M. Lucas [Lucas89], a pour objectif de permettre à un utilisateur de décrire, engendrer et étudier des objets appartenant à des univers de formes, à partir de la simple spécification d'une listes de propriétés ou de contraintes. Le modeleur déclaratif doit alors créer automatiquement les objets correspondant à la description ainsi fournie. Ce projet a, par la suite, donné naissance à plusieurs modeleurs concernant chacun un univers de formes particulier.

> **MultiFormes** [Plemenos91] : Ce projet s'intéresse à la création de scènes au moyen d'un langage, basé sur des règles PROLOG, permettant de décrire les objets élémentaires (cercle, triangle, carré, etc.) et de les relier entre eux (placement, copie de dimensions, etc.). Les différentes scènes, correspondant une même description, sont ensuite générées à l'aide de deux moteurs d'inférence, le premier gérant les règles constructives par chaînage avant, le second

gérant les règles destructives par chaînage arrière.

La multiplicité des solutions pose toutefois un problème d'interaction avec l'utilisateur (comment choisir rapidement une scène parmi celles qui sont solutions ?, laquelle présenter en premier ?, etc.).

VoluFormes [Chauvat94] : ce projet vise à construire un système permettant la description et la visualisation de paysages ou d'objets plus ou moins imaginaires. Les objets recherchés sont tout d'abord décrits à l'aide d'ébauches (ensembles de boîtes 3D), qui servent à déterminer grossièrement la forme recherchée. Il est donc nécessaire de pouvoir décrire les ensembles de volumes qui servent à construire une ébauche. Pour cela, l'utilisateur dispose d'un langage de description qui lui permet d'assembler les boîtes 3D les unes avec les autres.

Le système fait ensuite croître une forme particulière dans chacune des boites ; c'est ainsi que l'on passe de l'ébauche au résultat final.

PastoFormes [Colin90] : modélisation déclarative à base de collage de polyèdres élémentaires

AutoFormes [Martin90] : modélisation déclarative à base d'automates cellulaires,

PolyFormes [Martin93] : modélisation déclarative de polyèdres

CordiFormes [Desmontils95] : Plate-forme de développement de modeleurs déclaratifs

2.6.2 Les travaux de Donikian

Stéphane Donikian propose une méthode de conception déclarative, dénommée Scriptographie [Donikian93], basée sur la création d'un scénario décrivant l'environnement du point de vue d'un observateur. L'espace est décrit, depuis un point de vue donné, en termes de propriétés, de contraintes et de relations portant sur les objets :

« *Je vois un mur de 5 mètres de haut avec une petite fenêtre en son milieu* »

L'ensemble de ces descriptions, une par point de vue, est alors pris en compte puis retranscrit sous la forme de règles PROLOG. La scène virtuelle est ensuite reconstruite par inférence. L'objectif de ces travaux est de vérifier la cohérence de la description fournie par un architecte et d'en proposer une représentation graphique.

Comme tous les systèmes déclaratifs basés sur des moteurs d'inférences, les temps de calculs nécessaires à la génération d'une scène peuvent être extrêmement longs, pouvant aller de plusieurs heures à quelques jours.

3 Prototypage virtuel interactif

Nous avons réalisé, dans le chapitre précédent, un tour d'horizon des différentes techniques traditionnellement utilisées pour modéliser un environnement virtuel. Nous avons ainsi pu remarquer que ce type d'opération restait, encore de nos jours, une tâche longue et difficile, surtout lorsque l'utilisateur ne dispose pas d'un modèle réel à partir duquel il pourrait reconstruire la scène virtuelle. Pourtant, maîtriser ce type de problème et le rendre accessible à des utilisateurs non expérimentés représente un enjeu industriel stratégique. En effet, il est désormais devenu important de pouvoir construire, dès les phases d'avant projet, un prototype du produit à réaliser. Certes, un système de CAO pourrait être utilisé pour remplir ce type de tâche. Mais le manque de convivialité et d'interactivité de tels systèmes reste rédhibitoire quant à leur utilisation par des personnes novices en la matière.

Dans ce chapitre, nous allons nous intéresser aux techniques d'interaction tridimensionnelle appliquées à la construction d'environnements virtuels. Ces techniques, généralement regroupées sous le nom de réalité virtuelle, semblent pouvoir offrir un niveau d'ergonomie tel que l'on peut désormais imaginer des systèmes de prototypage virtuel pouvant être manipulés par des utilisateurs non experts.

Nous présenterons les principales expériences réalisées dans ce domaine puis nous définirons le projet PROVIS, conjointement lancé par le Centre National d'Etudes Spatiales et la société CISI afin d'étudier et de réaliser un système pour le prototypage virtuel coopératif. C'est dans le cadre de ce projet qu'ont été effectués les travaux présentés dans ce mémoire.

3.1 Périphériques pour l'interaction tridimensionnelle

Faire interagir l'homme avec un environnement virtuel constitue l'un des principes fondamentaux de la réalité virtuelle [Burdea93]. Il existe pour cela de nombreux périphériques dédiés à l'acquisition ou à la restitution des informations multisensorielles qu'est à même de traiter un système informatique.

3.1.1 Systèmes de localisation et d'orientation spatiale

Les dispositifs de suivi de position et d'orientation sont des périphériques essentiels dès lors que l'on souhaite immerger un utilisateur dans un environnement virtuel. Dans ce cas, il est par exemple nécessaire de connaître la position et l'orientation de la tête de l'utilisateur afin de générer les images correspondantes. Pour ce faire, il existe quatre types de technologies utilisables pour effectuer les mesures.

3.1.1.1 Principes de mesure

Mesure mécanique

Les dispositifs mécaniques mesurent la position et l'orientation d'un objet à l'aide d'un bras articulé [Figure 37] reliant le point de mesure à un point de référence. La mesure des angles en chaque articulation ainsi que la connaissance de la structure du bras permettent alors de calculer la position et l'orientation du point suivi.

Figure 37 : *Le système PHANToM 3.0 (intégrant un dispositif de retour d'effort)*

Mesure optique

Les dispositifs optiques utilisent généralement des marqueurs [Figure 38], placés sur l'objet ou l'utilisateur à suivre, qui sont ensuite repérés dans l'espace à l'aide d'un ensemble de caméras (cf. paragraphe 1.1.7.3).

Figure 38 : *Utilisateur muni de marqueurs*[1]

Mesure magnétique

Un tel système est généralement composé de trois éléments [Figure 39]:

La source magnétique : elle est constituée de trois bobines magnétiques, mutuellement perpendiculaires, qui émettent un champ magnétique lorsqu'elles sont traversées par un courant électrique.

Le récepteur : le récepteur est construit, de manière identique, à partir de trois bobines qui génèrent un courant électrique lorsqu'elles sont placées dans un champ magnétique.

Le boîtier de contrôle : cet appareil est en charge du fonctionnement du système complet. Pour cela, il pilote la source magnétique en lui faisant émettre, à tour de rôle, trois champs magnétiques perpendiculaires deux à deux. Un champ émis induit un courant dans les bobines du récepteur dont l'intensité dépend de la distance à laquelle il se trouve par rapport à la source. A partir de l'ensemble des courants générés, le boîtier de contrôle détermine la position et l'orientation du récepteur.

[1] Photographie Acclaim Sports

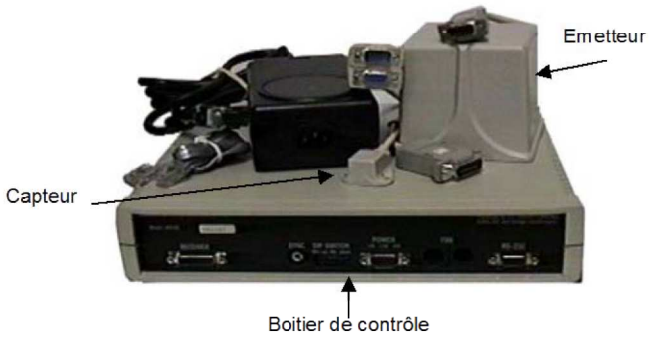


Figure 39 : *Le système Flock of Birds d'Ascension Technology Corp.*

La qualité de la mesure d'un tel dispositif, en termes de précision et de rapidité, est tout à fait convenable dans un rayon de quelques mètres autour de l'émetteur. En revanche, ce type de système, au demeurant d'un faible encombrement, possède un désavantage de taille : il ne peut être utilisé dans un environnement comportant des objets métalliques. Ces derniers sont en effet susceptibles de réfléchir les ondes magnétiques et de perturber ainsi les mesures.

Mesure acoustique

Il existe deux méthodes de mesure acoustique. La première consiste à émettre un signal ultrasonore depuis une source et à mesurer le temps que met un récepteur à le capter. Lorsque l'on utilise trois sources, émettant chacune à une fréquence propre, et trois récepteurs, on peut déterminer la position et l'orientation d'un objet [Figure 40]. La deuxième méthode est similaire à la première à ceci près qu'elle compare la phase du signal émis par rapport à la phase d'un signal de référence.

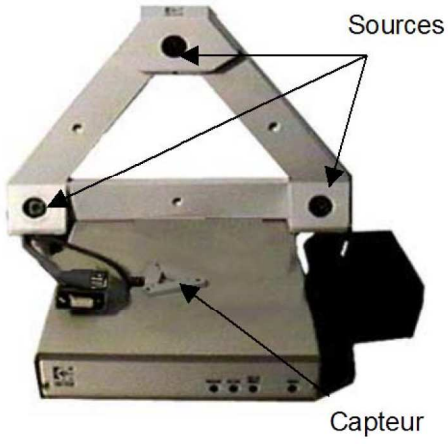


Figure 40 : *Le système de localisation acoustique de Logitech*

Evaluation

Nous présentons [Tableau 1] l'évaluation [Meyer92] réactualisée des technologies présentées dans ce paragraphe en fonction des critères suivants :

Précision, résolution : la résolution détermine la plus petite valeur que le système peut mesurer. Des variations de position ou d'orientation inférieures à la résolution ne seront pas détectées par le système. La précision correspond à l'erreur de mesure.

Réponse : la réponse est déterminée par la fréquence d'échantillonnage, c'est à dire le nombre de mesures communiquées à l'ordinateur par seconde, et le temps séparant l'instant de la mesure de celui de la communication.

Robustesse : la robustesse correspond à la résistance du dispositif aux perturbations provenant de l'environnement de travail.

Sociabilité : la sociabilité du système est déterminée par la taille de l'espace de travail et par son adaptation à la poursuite de plusieurs objets dans cet espace.

Système	Précision	Réponse	Robustesse	Sociabilité	Commentaires
Mécanique	Bonne	Bonne	Bonne Faible sensibilité à l'environnement.	Encombrement assez important.	Bien adapté pour le retour d'effort Application en télérobotique.
Optique	Bonne La précision et la résolution diminuent quand le volume de travail augmente.	Bonne Ces systèmes sont bien adaptés au temps réel.	Bonne Certains systèmes peuvent être affectés par la lumière ambiante.	Sensible au masquage des marqueurs ou des caméras. Encombrement assez important.	Peut demander des installations complexes pour obtenir de bons résultats.
Magnétique	Bonne dans les petits espaces de travail. Dépend fortement de la présence de métal dans la zone de travail.	Fréquence d'échantillonnage relativement faible.	Grande sensibilité au métal dans la zone utile.	Bonne dans les petits espaces de travail.	Le plus couramment utilisé. Possibilités de suivre plusieurs objets simultanément en rajoutant des capteurs au système.
Acoustique	Bonne.	Dépendant de la distance	Dépendant de la distance	Sensible au masquage des capteurs ou des sources.	Extrêmement stable dans la zone utile. Peu onéreux et peu encombrant.

Tableau 1 : *Evaluation des technologies de poursuite de position et d'orientation [Meyer92]*

3.1.2 Contrôleurs 3D

Les contrôleurs 3D constituent une classe de périphériques dont la fonction première est de permettre à un utilisateur de manipuler un objet, ou tout autre composant de l'environnement virtuel. Contrairement aux périphériques de suivi décrits précédemment, les contrôleurs 3D restent fixes par rapport à l'espace de travail et permettent le contrôle incrémental d'une position ou d'une orientation. Ils fonctionnent donc en mode relatif. Ce type de dispositif [Figure 41] est généralement constitué d'une sphère ou d'un palet de contrôle, plus ou moins fixe par rapport au socle de l'appareil, ainsi que d'un mécanisme [Hirzinger92] composé de 6 LED et de 6 capteurs photosensibles permettant de mesurer les forces et les couples qui lui sont appliqués. De plus, un certain nombre de boutons de contrôle sont disponibles sur l'appareil, à portée de main.

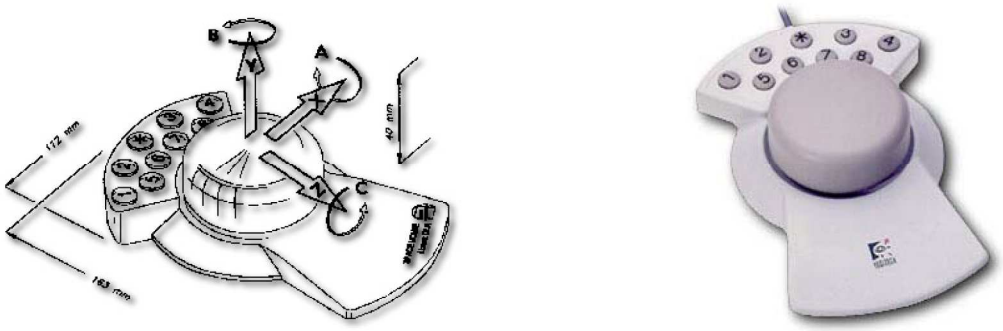


Figure 41 : *Contrôleur SpaceMouse de DLR (vendu sous le nom de Magellan par Logitech)*

La métaphore d'interaction proposée consiste à identifier la sphère ou le palet du contrôleur à un objet que l'utilisateur peut prendre en main. Il lui est alors possible d'imposer des contraintes de mouvement afin de déplacer ou d'orienter cet objet dans l'espace. Ce type de périphérique offre une position d'utilisation confortable puisque l'avant bras de l'utilisateur peut reposer sur la table. Il nécessite cependant une période d'entraînement de quelques heures, notamment pour pouvoir contrôler indépendamment chaque degré de liberté.

3.1.3 Les gants de données

Ce type de périphérique permet de mesurer les mouvements des doigts de la main [Sturman94]. Il est généralement associé à un système de suivi magnétique afin de repérer la position et l'orientation générale de la main. Les principes que nous décrivons ci-dessous nécessitent tous une phase de calibration afin de prendre en compte la morphologie de la main de l'utilisateur ainsi que les différentes variations de positionnement des capteurs d'une session de travail sur l'autre.

3.1.3.1 Principes de mesure

Mesure mécanique

Ce type de dispositif n'est pas à proprement parlé un gant de données. Il s'agit plutôt d'une structure articulée [Figure 42] que l'utilisateur installe sur sa main. Chaque angle articulaire est mesuré à l'aide d'un capteur placé aux articulations de la structure mécanique.

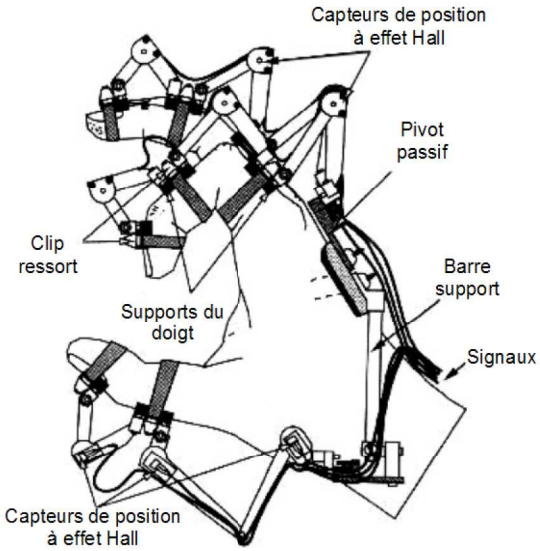


Figure 42 : *Le Dextrous Hand Master d'Exos Inc. (extrait de [Burdea93])*

De nos jours, cette méthode n'est pratiquement plus utilisée pour la seule analyse des mouvements des doigts. Elle est par contre parfois associée à un dispositif de retour d'effort.

Mesure optique

Le principe consiste à disposer une fibre optique sur chaque doigt. Une diode est montée à une extrémité de cette fibre et un phototransistor mesure la lumière à l'autre extrémité. La flexion du doigt modifie la transmission de la lumière à l'intérieur de la fibre, permettant ainsi une estimation de l'angle de flexion. On utilise en général plusieurs fibres optiques par doigt afin de déterminer l'angle de flexion pour chaque phalange.

Ce type de dispositif, relativement précis et d'un encombrement raisonnable, souffre néanmoins d'un manque chronique de fiabilité dû à la fragilité des fibres optiques et à la sensibilité du dispositif de mesure qui doit rester parfaitement en place pour être fonctionnel.

Mesure par jauge de contrainte

Le principe consiste à intégrer au gant des petites lamelles, placées sur chaque articulation, qui permettent de mesurer la flexion de chaque phalange. Il existe plusieurs types de lamelles qui font office de jauges de contraintes (capteurs à encre conductrice pour le PowerGlove de Mattel, matériau élastique à base de Nylon pour le CyberGlove [Figure 43]).

Figure 43 : *Les gants de données CyberGlove de la société Virtual Technologies*

Le système proposé par Virtual Technologie constitue sans doute la meilleure offre en terme de gant de données sans retour d'effort. Précis, fiable et d'un encombrement minimal, le CyberGlove ressemble réellement à un gant traditionnel. De plus, la faible taille des capteurs permet de les disposer (jusqu'au nombre de 24) un peu partout sur la main, autorisant ainsi une mesure extrêmement complète des mouvements (adduction/abduction entre chaque doigt, flexion du poignet).

3.1.4 Systèmes à retour tactiles et d'efforts

Les systèmes à retour d'effort permettent de restituer une force en réponse aux manipulations de l'utilisateur. Ils constituent une source d'information tout à fait essentielle dans un processus d'interaction. Il est à ce sujet paradoxal d'imaginer pouvoir interagir avec un environnement virtuel, comme on le ferait en réalité, sans ressentir de sensations tactiles ou d'efforts. Le lecteur intéressé par ce sujet pourra se reporter à [Burdea93].

Retour tactile

Le retour tactile permet de fournir des informations sur la surface de l'objet manipulé (température, rugosité, viscosité, planéité de la surface). Il existe plusieurs mécanismes, généralement intégrés à un gant de données, pour restituer ce genre de sensation.

Le gant Teletact [Stone91] utilise par exemple des petites pastilles gonflables qui permettent d'exercer ou de relâcher une pression en différents endroits de la main. Le gant CyberTouch de Virtual Technologies propose, quant à lui, des vibreurs situés sur chaque doigt. Ce dispositif n'est pas vraiment très pertinent dans la mesure où les vibrations générées par le contact prolongé avec un objet sont assez désagréables et peu convainquantes. Le système fonctionne par contre relativement bien lorsqu'il s'agit de simuler des contacts très brefs.

Figure 44 : *Le système CyberTouch de Virtual Technologies*

Retour d'effort

Le retour d'effort permet au système de restituer une force en réponse à une action de l'utilisateur. Il existe une variété de périphériques intégrant un dispositif de retour d'effort. Nous pouvons par exemple citer les bras maîtres pour la téléopération, les manches à balai du type ROBOTOP [Bois96], les souris à retour d'effort de Haptic Technologies ou encore le clavier Retroactif Modulaire de l'IMAG [Cadoz90].

Plusieurs technologies peuvent là encore être utilisées pour générer une force sur la main de l'utilisateur. On peut ainsi utiliser un système de moteurs [Bois96] [Figure 45], de micro-vérins pneumatiques [Burdea93] ou de câbles qui passent sur le dos de la main [Figure 46] [Bouzit93]. Ces deux derniers systèmes n'offrent cependant pas la possibilité de ressentir le poids des objets puisqu'ils s'appliquent aux doigts de la main.

Figure 45 : *Les systèmes PHANToM*

Figure 46 : *Le CyberGrasp de Virtual Technologies*

3.1.5 Périphériques pour la visualisation stéréoscopique

Dans une application 3D, la perception de la profondeur, c'est à dire la distance au point de vue, est une caractéristique essentielle pour comprendre correctement la structure spatiale de l'environnement virtuel affiché. La dimension de profondeur est d'autant plus difficile à appréhender que le rendu proposé est bien souvent monoscopique et donc purement bidimensionnel. Cependant, plusieurs informations peuvent être utilisées pour percevoir des informations de profondeur :

Occlusion : les objets lointains sont masqués par les objets plus proches de l'observateur.

Perspective : les objets lointains apparaissent plus petits que les objets proches, le changement d'échelle étant proportionnel à la distance au point de vue. En fait, la distance perçue est liée de manière linéaire à la taille de l'objet affiché (loi de Emmert).

Effets d'illumination : les effets d'éclairage et d'ombrage permettent la reconstruction mentale de formes tridimensionnelles. Il est à noté que l'ombre portée des objets, générée par les sources de lumière, apporte un supplément d'informations. Cependant, leur génération, très gourmande en temps de calcul, n'est pratiquement jamais supportée par les systèmes graphiques temps réel.

Toutes ces informations ne suffisent cependant pas à percevoir correctement la dimension de profondeur d'un objet. Ce sont pourtant les seules qu'il soit possible de fournir avec une seule image. En réalité, le cerveau utilise des facteurs plus complexes pour appréhender la profondeur et donc le relief de l'univers :

Disparité binoculaire : les images de l'environnement sont perçues par les deux yeux simultanément. Le cerveau dispose donc de deux images prises depuis deux points de vue légèrement décalés ; la distance les séparant est appelée distance interpupillaire.

Parallaxe : l'effet de parallaxe est lié au changement de position apparente d'un corps, dû à un changement de position de l'observateur : les objets les plus proches semblent alors se déplacer plus que les objets éloignés.

Convergence : la convergence est la rotation simultanée des yeux lors de la mise au point sur un objet plus ou moins proche.

Accommodation : il s'agit là de l'ajustement de la distance focale du cristallin lors de la mise au point.

Il est possible en pratique de restituer ces effets. Cependant, leur génération est souvent source de contraintes et de coûts en calculs additionnels non négligeables. En effet, le principe de base de la restitution graphique en relief repose sur la génération, non plus d'une image, mais d'un couple stéréoscopique d'images. Cette opération nécessite un temps de calcul quasiment doublé et l'utilisation de matériel de visualisation spécifique comme, par exemple, des lunettes à obturateurs, un casque de visualisation ou encore une dalle polarisante.

La restitution de l'effet de parallaxe nécessite, quant à elle, la connaissance en continu des mouvements du point de vue de l'observateur ainsi qu'un rendu suffisamment rapide pour que les images produites soient constamment en accord avec celui-ci. Ceci accentue les contraintes de performance puisqu'une fréquence d'affichage inférieure à 5 images par seconde n'offre aucune illusion d'animation [Bryson93]. Pire, l'utilisateur peut rencontrer certains troubles physiologiques [Kennedy93], comme des sensations de fatigue visuelle, de nausée ou des troubles oculomoteurs, lorsqu'on lui présente des images en relief à une fréquence insuffisante [Paush92].

Le facteur de convergence peut, quant à lui, être mesuré à l'aide d'un système d'analyse du regard appelé oculomètre. Ce dispositif, généralement composé d'une mini-caméra et d'une diode, permet de connaître la position de la pupille par rapport à la sclérotique visible. Outre son encombrement, l'inconvénient de ce genre de système est qu'il doit resté fixe par rapport à l'œil qu'il analyse. En d'autre termes, un système d'oculométrie est utilisable soit installé sur la tête de l'utilisateur (il peut par exemple être intégré à des lunettes) soit si la tête de ce dernier reste absolument immobile. En conclusion, il est extrêmement difficile de prendre en compte le facteur de convergence. C'est pourquoi l'on fixe un point de convergence à une distance moyenne rendant la génération d'un relief correct impossible lorsque l'utilisateur regarde un objet situé plus près ou plus loin que cette position.

3.1.6 Casque de visualisation stéréoscopique

Afin de restituer les informations nécessaires à la perception d'une image en relief, les casques de visualisation stéréoscopiques intègrent les dispositifs suivants :

Affichage : un petit dispositif d'affichage (généralement un écran à cristaux liquides) est placé devant chaque œil.

Optiques : une lentille est disposé devant chaque écran afin d'accroître le champ de vision et de corriger, éventuellement, la distorsion de l'écran.

Repérage : un système de suivi de la position et de l'orientation de la tête est généralement intégré dans ce type d'appareil. Il permet la génération des images en accord avec la position de l'observateur.

Deux types de casques sont disponibles sur le marché en fonction du type d'application visée. Pour les applications nécessitant l'immersion totale de l'utilisateur, il est possible d'utiliser un casque clos [Figure 47] où seules les images générées sont visibles. Pour d'autres applications, nécessitant par exemple la préservation d'un contact visuel entre l'utilisateur et son environnement, il est possible d'utiliser un casque non immersif où les images générées sont indirectement projetées sur des optiques transparentes placées devant les yeux de l'utilisateur [Figure 48]. On obtient ainsi des images synthétiques qui viennent se superposer au monde réel, ce principe étant généralement appelé réalité augmentée.

Figure 47 : *Casque clos ProView 60 de Kaiser*

Figure 48 : *Casque ouvert SIM EYE 40 de Kaiser*

La faible résolution d'affichage (généralement 640*480 pixels) et le champ de vision limité (en moyenne 40° horizontalement pour 30° verticalement) des casques de visualisation constituent leurs principaux inconvénients. Kaiser Electro Optics propose, à ce sujet, la juxtaposition de deux à six écrans par œil afin d'accroître résolution (1920*960) et champ de vision (180°H*90°V), le prix de cet appareil étant en rapport avec ses performances. Ce type de solution nécessite en effet la mise en place d'un système très coûteux de lentilles afin de produire une image cohérente à partir des écrans juxtaposés.

Les casques de visualisation sont généralement mal adaptés à des travaux ponctuels ou en groupe. Ils nécessitent un certain temps d'installation ainsi qu'une zone de travail assez large. On réservera donc leur utilisation à des domaines d'applications très particuliers (simulations, formation) nécessitant un haut degré d'immersion. Celui-ci se fera au détriment des coûts matériels et de la qualité d'affichage qui, bien souvent, ne permet pas de lire les textes affichés à une taille normale.

Il est à noter que la société FakeSpace propose un dispositif hybride (BOOM) composé de deux écrans CRT que l'utilisateur dispose devant son visage. Ce dispositif est maintenu en l'air par un bras articulé qui permet le suivi du dispositif dans l'espace. Il ne s'agit pas à proprement parler d'un casque mais plutôt d'un système d'écrans que l'utilisateur peut déplacer à l'aide de deux poignées. L'avantage principal de ce dispositif est d'offrir une résolution similaire à celle d'un moniteur d'ordinateur.

3.1.7 Lunettes stéréoscopiques

L'utilisation des lunettes stéréoscopique permet d'améliorer les capacités de perception de la profondeur dans des configurations non immersives. Les principaux avantages de telles lunettes sont leur faible encombrement, leur coût modéré, la disponibilité de la surface entière de l'écran pour l'affichage ainsi que la possibilité de rester en contact avec l'environnement de travail. De plus, ce type de système peut être rapidement installé. Il existe deux types de technologies utilisables.

Lunettes actives

Pour pouvoir utiliser ces lunettes, la station graphique doit supporter un mode vidéo spécial où l'écran affiche, alternativement et à haute vitesse (120 Hz), les images destinées à l'œil droit et celles destinées à l'œil gauche. Les lunettes sont synchronisées avec l'écran à l'aide d'un petit boîtier qui leur indique l'œil auquel est destinée l'image en cours d'affichage, le verre situé devant l'autre œil est alors rendu opaque à l'aide de l'écran à cristaux liquides qu'il contient.

Figure 49 : *Lunettes CrystalEyes de StereoGraphics*

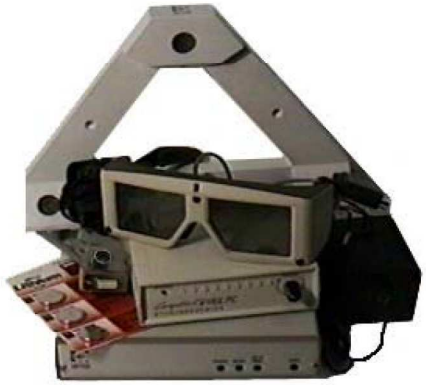


Figure 50 : *Système CrystalEyes VR de Stereographics*

Lunettes passives

Le principe consiste à polariser la lumière émise par le dispositif d'affichage (écran ou projecteur) en installant, par dessus, une dalle polarisante. Cette dalle est en charge d'alterner la polarité de la lumière émise en fonction de l'œil auquel est destinée l'image. Les verres des lunettes sont en fait des filtres qui ne laissent passer qu'un seul type de lumière (polarisée horizontalement ou verticalement). Cette solution, relativement onéreuse pour quelques personnes (le coût de la dalle est élevé), devient vite rentable dès lors qu'il s'agit de faire participer de nombreux utilisateurs simultanément, le coût d'une paire de lunette étant dérisoire (quelques francs).

3.2 Interaction tridimensionnelle

Il est de plus en plus évident que la traditionnelle interface homme machine en deux dimensions, et les métaphores d'interaction qui lui sont associées, sont devenues inadaptées pour les applications nécessitant la spécification d'informations spatiales complexes. En particulier, le faible débit de communication entre l'utilisateur et l'application, inhérent à l'utilisation traditionnelle d'une souris pour l'interaction 3D, compliquent considérablement le travail des utilisateurs [Conner92] [Gobbetti93] [Gobbetti95.1].

Le retour fourni à l'utilisateur pose lui aussi un problème : le manque d'informations quant à la structure de l'environnement modélisé, représenté sur une image fixe, oblige souvent à utiliser plusieurs vues afin d'obtenir des informations complémentaires. Il est alors demandé à l'utilisateur de combiner ces différentes vues pour former le modèle mental du prototype sur lequel il travaille. Ce processus complique encore un peu plus un travail déjà difficile [Herdon92]. En outre, il oblige l'utilisateur à se concentrer sur la manière d'obtenir un résultat et non sur le travail à réaliser.

C'est pourquoi il paraît nécessaire de repenser les métaphores de l'interface homme machine afin de permettre aux utilisateurs de travailler directement en trois dimensions lorsque cela s'avère nécessaire. En partant du postulat selon lequel l'être humain est parfaitement capable d'interagir avec le monde dans lequel il vit, la recherche en réalité virtuelle propose des solutions permettant à un utilisateur d'interagir avec des mondes virtuels de la même façon qu'il interagirait avec le monde réel. C'est ainsi que l'on devrait pouvoir rendre l'interaction beaucoup plus naturelle et donc plus facile à appréhender.

Cependant, les difficultés liées à l'immersion d'un utilisateur dans un environnement virtuel ont conduit les chercheurs en environnements virtuels à se concentrer d'avantage sur le développement de nouveaux périphériques d'entrée et de sortie, présentés dans le chapitre précédent, que sur de nouveaux paradigmes pour l'interaction 3D.

Les recherches dans le domaine de l'interaction homme machine se sont, quant à elles, d'abord orientées vers l'exploration de nouvelles interfaces pour le travail en trois dimensions en utilisant de nouveaux paradigmes, certes intuitifs, mais ne tirant pas profit des nouveaux périphériques de réalité virtuelle [Robertson93] [Herndon94] [Gobbetti95.2]. C'est ainsi que l'on a vu apparaître des éléments d'interface (widgets) tridimensionnels, plus ou moins utiles, que l'on actionne à la souris [Figure 51].

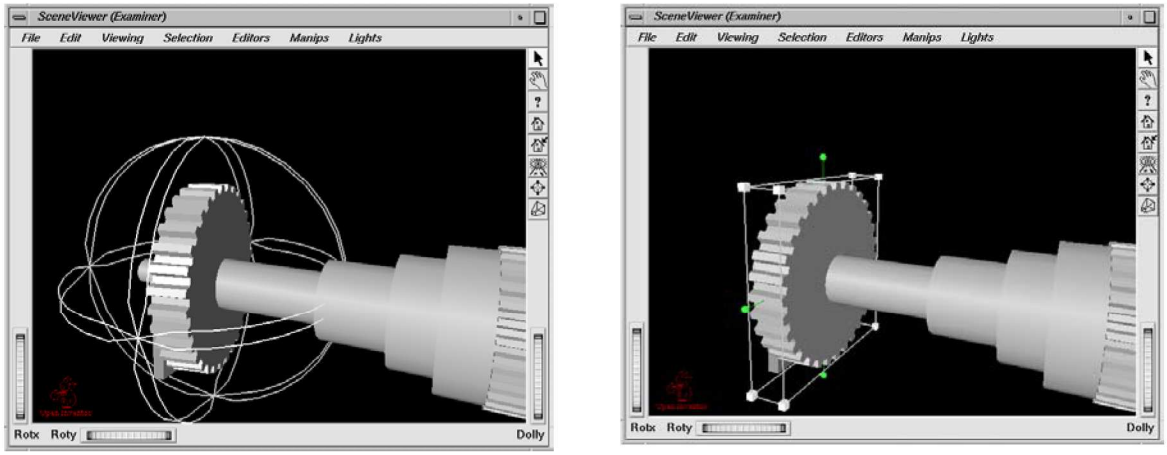


Figure 51 : *Deux exemples de manipulateurs 3D d'Inventor [Wernecke95]*

Le modeleur développé par AT&T [Weiner89] fut pourtant l'une des premières applications révélant l'importance de l'utilisation simultanée de plusieurs périphériques pour le travail en trois dimensions. Cependant, son interface utilisateur était limitée à l'affichage de menu en 3D et n'autorisait que le déplacement et l'orientation d'objets. 3-Draw [Sachs90] et le modeleur polygonal décrit dans [Shaw94] ont étendu ces techniques d'interaction en permettant l'utilisation simultanée des deux mains pour effectuer ces opérations ; cette technique fut d'abord appliquée par Buxton et Myers [Buxton86] qui utilisent deux souris pour piloter une interface 2D. Les systèmes de Sachs proposent deux capteurs magnétiques, de type Polhemus, qui permettent à l'utilisateur de manipuler simultanément un curseur 3D ainsi que l'orientation de la surface de travail. Il est alors nécessaire de corréler visuellement les mouvements de l'utilisateur avec leur effet dans l'environnement virtuel. Cette opération, assez simple à réaliser lorsque l'on utilise un casque de visualisation, devient délicate à mettre en œuvre lorsque l'on affiche les images sur un écran. Les systèmes UGA [Zeleznik91], de l'université de Brown, et VB2 [Gobbetti93], de l'institut fédéral suisse de technologie de Lausanne, ont permis de démontrer la souplesse et la puissance des widgets 3D, associées à un affichage stéréoscopique sur écran, lorsque l'on corrèle correctement les manipulations de l'utilisateur avec les effets produits sur les objets contrôlés.

La plupart des systèmes développés utilisent des périphériques à repérage absolu. C'est le cas de Holosketch [Deering95] qui propose à la fois le suivi de la tête et de la main de l'utilisateur, à l'aide de capteurs magnétiques, afin de rendre encore plus interactive la modélisation de formes tridimensionnelles. Deering observe que l'utilisation de tels dispositifs entraîne, relativement rapidement, une sensation de fatigue chez l'utilisateur qui a constamment les mains en l'air. Le système Leman [Turner95] a démontré comment une interface [Turner96] basée sur un suivi absolu de la tête et la mise à disposition d'un contrôleur 3D, incrémental, pouvait être utilisée, avec succès et sans engendré de sensation de fatigue, pour la création de personnages animés.

3.3 Prototypage virtuel interactif

Dans les projets d'ingénierie à grande échelle, l'étape du design est souvent la plus délicate car les choix pris durant cette phase peuvent par la suite altérer considérablement le résultat final ainsi qu'être source de pertes de temps et d'argent. Le prototypage virtuel constitue une solution permettant aux décideurs et aux ingénieurs de tester et d'améliorer, plus efficacement, le design d'un produit en remplaçant les maquettes physiques par des maquettes virtuelles [Balaguer96.1].

Ces prototypes sont actuellement construits à l'aide d'outils de CAO (CATIA, EUCLID, Pro-Engineer) dont les capacités sont pourtant trop limitées pour permettre une navigation et une manipulation interactive des objets modélisés. Avec de tels systèmes, il faut beaucoup de temps et d'imagination pour déceler les erreurs de design. Ces outils ont aussi pour but de fournir une description extrêmement précise des objets modélisés. En effet, c'est le modèle généré par les outils de CAO qui reste, de nos jours, la référence tout au long du processus de création d'un produit, depuis sa conception jusqu'à sa fabrication. C'est aussi pour cette raison que ces outils sont mal voire pas du tout adaptés au temps réel. En effet, la précision des modèles qu'ils produisent rend tout simplement impossible leur manipulation direct dans un système de réalité virtuelle.

De plus, il est nécessaire de disposer d'une grande expertise pour pouvoir utiliser un système de conception assistée par ordinateur. Cependant, les décideurs et les concepteurs, impliqués dans les choix stratégiques et techniques dès les phases d'avant projet, ne disposent généralement pas d'une telle expertise ; ce sont pourtant leurs choix qui orienteront le projet.

C'est pourquoi de grands projets d'ingénierie ont souvent nécessité le développement spécifique d'outils de prototypage virtuel interactifs [Ellis96] utilisant les techniques d'interaction 3D décrites dans le chapitre précédent. Nous pouvons citer, par exemple, le ISS VR Demonstrator utilisé par Rolls-Royce pour vérifier le concept de prototypage et de maintenance de moteurs virtuels [Greenfield96], le performant système FlyThru de Boeing, conçu pour le design du 777 [Abarnabel96], et enfin le système i3d développé conjointement par le CERN et le CRS4[1] pour l'aide à la conception et à la construction du Large Hadron Collider du CERN [Balaguer96.1]. Nous présentons plus en détails, ci-après, ces deux derniers projets car ils constituent sans aucun doute les projets de prototypage virtuel interactif les plus importants.

[1] Centre for Advanced Studies, Research and Development in Sardinia

3.3.1 Le système FlyThru de Boeing

A la fin des années 80, la compagnie Boeing décida de construire le 777, un bi-moteur à gros portage. Le choix fut alors fait de concevoir cet avion sans réaliser les traditionnelles maquettes physiques de classe 3 qui grèvent considérablement le coût d'un tel programme. Pour la première fois, un avion allait être entièrement conçu à l'aide d'un système de conception assistée par ordinateur, à savoir le produit CATIA de Dassault Systèmes. En 1989, environ 5000 modèles numériques avaient déjà été réalisés. Aujourd'hui, le projet 777 comporte plus de 40000 modèles nécessitant plus de 15 giga-octets de mémoire pour leur stockage. Il était évident, en 1990, que les systèmes de CAO ne pouvaient permettre de visualiser que 10 à 20 de ses modèles simultanément, soit 100 fois moins que nécessaire aux designers et autres décideurs pour maîtriser le bon déroulement du programme.

En 1992, le département Boeing Computer Services mit à la disposition des concepteurs une application, développée en interne, du nom de FlyThru. Cette application, dédiée à la visualisation de grandes bases de données, fut rapidement adoptée au point d'être aujourd'hui présente dans quasiment tous les secteurs d'activité de Boeing consacrés à la construction d'avions civils ou militaires.

La principale caractéristique de FlyThru est de permettre la visualisation interactive et simultanée de plus d'un millier de modèles numériques complexes issus de la CAO. De plus, ce système permet de gérer toutes sortes de données et de services connexes. Il est par exemple possible d'utiliser FlyThru pour la gestion de la documentation des composants, pour l'analyse des anomalies de conception ou encore pour la revue de projet, avec des utilisateurs distants, au moyen de connexions réseau. C'est ainsi que, pour la première fois, la totalité des différentes documentations relatives à un avion purent être réalisées et livrées aux clients avant la livraison des appareils.

Afin d'aider les milliers d'ingénieurs et de décideurs impliqués dans un programme de l'ampleur de celui du 777, un système de distribution des données a été créé sous le nom de « Update ». Ce système permet de rendre accessible, à tout instant, n'importe quel modèle depuis n'importe quelle station FlyThru. Pour ce faire, les versions des modèles sont transférées chaque soir vers le serveur FlyThru d'où, ensuite, elles sont diffusées vers les utilisateurs désirant utiliser l'application.

L'aspect le plus pointu de ce système réside dans ses performances graphiques. Il est en effet quasiment impossible de traiter, tel quel, une base de modèles de la taille de celle créée pour le développement d'un avion. C'est pourquoi FlyThru intègre des algorithmes d'optimisation (« occlusion culling », « niveaux de détails ») [Brechner95] pour le rendu temps réel de grandes bases de données.

Grâce à la technologie de FlyThru, la conception du Boeing 777 a été largement facilitée. En effet, les erreurs fréquemment rencontrées sur d'autres types d'avions furent dans une large mesure évitées. En outre, le programme du 777 fut mené à bien plus rapidement, avec plus de précision, tout en coûtant moins cher que n'importe quel autre précédent programme de Boeing. Selon les statistiques, FlyThru a optimisé le design de l'avion et a permis l'économie de plusieurs dizaines de millions de dollars. De nombreux autres programmes, tel que ceux du F22 et de l'AWACS, ont confirmé depuis les bénéfices du « tout numérique ».

3.3.2 Le projet VENUS du CERN

Le projet VENUS consiste à développer une solution pour le prototypage virtuel du Large Hadron Collider (LHC) qui constituera, une fois construit en 2004, le plus grand accélérateur de particules au monde. Ce système, pour des raisons de coût de fabrication, doit partager les 27 kilomètres de tunnels qu'emprunte le système Large Electron Positron collider (LEP) déjà en place. Le LHC est un projet à très grande échelle impliquant plus de 300 intervenants industriels. Il est donc crucial de pouvoir maîtriser et analyser très précisément chaque détail de ce projet afin d'en maîtriser le coût et le temps de développement.

Comme nous l'avons déjà noté précédemment, les systèmes de CAO ne sont pas suffisamment interactifs pour permettre l'analyse rapide et intuitive de l'infrastructure d'un modèle 3D. Ils sont, de plus, difficiles à utiliser pour une personne non experte. C'est pour ces raisons qu'en 1994 fut lancé le projet VENUS. Afin de répondre aux besoins des décideurs et des concepteurs, plusieurs fonctionnalités furent alors identifiées. Le système développé devait pouvoir permettre la visualisation interactive d'un prototype virtuel, le partage d'informations au travers d'un réseau informatique ainsi que l'étude de l'impact de la construction du LHC sur l'environnement.

C'est ainsi que furent spécifiées les fonctionnalités du système i3D, développé à l'origine par le CRS4 [Balaguer95.2] [Gobbetti95.2] pour la visualisation de scènes VRML. Cet outil propose une interface [Figure 52] de type « desktop VR », composée d'un contrôleur 3D, d'une souris et de lunettes stéréoscopiques pour la vision en relief. Le contrôleur 3D est utilisé, selon la métaphore « eye-in-hand » [Ware90], pour le contrôle de la position et de l'orientation de la caméra au sein de l'environnement virtuel. La souris est quant à elle destinée à la sélection d'objets et à l'accès d'informations multimédia associées aux objets 3D.

Figure 52 : Configuration matérielle du système i3D

Les composants 3D sont construits à l'aide du logiciel de CAO EUCLID puis convertis au format VRML. Le logiciel EUCLID utilisant une représentation CSG pour la définition des modèles, il est alors possible de générer automatiquement différents niveaux de détails pour chacun des objets convertis afin d'optimiser le rendu graphique.

L'utilisation du format VRML permet à i3D de traiter un prototype dont les différents composants sont situés à un endroit quelconque du réseau. C'est ainsi que chaque équipe peut réaliser sa partie du prototype et la rendre disponible sur le réseau interne du CERN. Lors d'une revue de projet, le logiciel i3D retrouve les différents composants et présente le prototype dans sa globalité. Un responsable pourra alors consulter la documentation associée à un composant en activant le lien hypertexte qu'il contient. Il s'agit d'une méthode de coopération asynchrone où un utilisateur met les résultats de son travail à la disposition d'autres personnes pour une utilisation ultérieure.

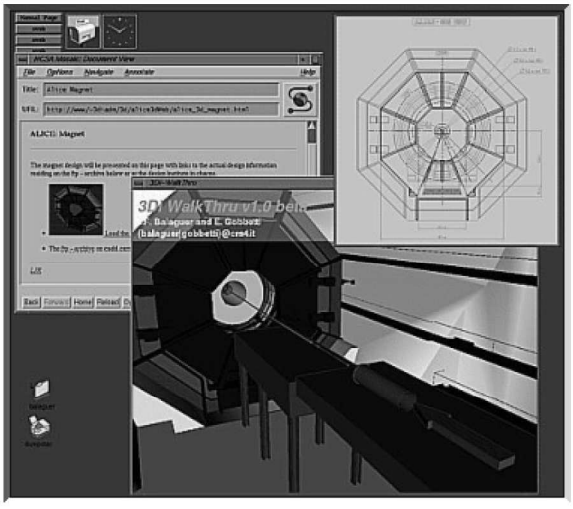


Figure 53 : *Utilisation du prototype 3D pour accéder à des informations multimédias distribuées*

Le projet VENUS était à l'origine destiné à fournir un système limité à la visualisation interactive de prototypes virtuels. Cependant, les récents travaux du CERN tendent à prouver que le système i3D peut aussi rendre de nombreux services pour la planification des opérations de construction et d'assemblage des différents composants du LHC. Celui-ci contiendra, par exemple, le sous-système ATLAS composé de plusieurs pièces pesant chacune 200 tonnes. Les différentes pièces seront descendues par un ensemble de grues, dans un puits, pour être ensuite mises en place dans le LHC. Ces opérations doivent respecter des contraintes de manipulation et de précision extrêmement fortes : l'accélération maximale ne doit pas dépasser 0,1 G lors des déplacements et la précision avec laquelle les pièces doivent être mises en place est de l'ordre de 6 microns. Ces opérations ne pouvant être improvisées, le projet VENUS s'oriente désormais vers l'utilisation d'i3D pour la planification de la procédure de mise en place d'ATLAS.

C'est ainsi qu'i3D permet désormais la définition de comportements qui peuvent être associés aux objets. Les premières études ont porté sur la planification des opérations que les grues devaient effectuer pour permettre l'installation d'ATLAS en toute sécurité. C'est ainsi que l'on a pu démontrer que le diamètre du puits de descente d'ATLAS dans le LHC pouvait être réduit de 25 à 18 mètres. Cette réduction implique un certain nombre de manœuvres qui seront réalisées par l'action simultanée de deux grues.

Le projet VENUS étudie actuellement un système permettant la formation et l'entraînement des grutiers. Ce système sera du même type que ceux développé par Caterpillar [Figure 54] pour le pilotage d'engins de chantier ou par CISI [Figure 48] pour l'entraînement et l'évaluation des conducteurs de véhicules.

Figure 54 : Simulateur de conduite d'engins Caterpillar

Figure 55 : Simulateur de conduite de véhicules CISI

3.4 Le projet PROVIS

Le projet de recherche PROVIS (Prototypage Virtuel de Systèmes) [Duchon96] [Torguet97] [Balet97], initié en 1995 par le Centre National d'Etudes Spatiales, vise à étudier de nouvelles solutions pour simplifier le travail des décideurs et des concepteurs de systèmes qui, dès les phases d'avant projet, doivent disposer d'outils leur permettant d'appréhender et de vérifier intuitivement la pertinence de leurs choix (encombrement des structures, accessibilité des composants, etc.).

L'objectif premier était d'étudier les techniques d'interaction homme machine 3D appropriées pour une utilisation du système pouvant durer plusieurs heures. Nous avons pour cela choisi une interface matérielle du type « desktop VR » similaire à celle utilisée par le système i3D. Nous lui avons adjoint un système de suivi de la tête de l'utilisateur, intégré aux lunettes de visualisation stéréoscopique, afin de pouvoir générer les effets de parallaxe. Ce type de configuration permet de travailler longtemps sans engendrer de fatigue musculaire notable ; seule la vision stéréoscopique prolongée peut entraîner une gêne visuelle à la longue. L'utilisation d'un contrôleur 3D à retour d'effort aurait sans aucun doute constitué un atout supplémentaire. Malheureusement nous ne disposions pas d'un tel système pour nos travaux.

Une première liste de fonctionnalités fut spécifiée avec les utilisateurs finaux du département mini-satellites. Notre système devait ainsi pouvoir permettre :

- La gestion d'une bibliothèque de prototypes et de composants.
- L'assemblage interactif de ces composants à l'aide de liaisons mécaniques.
- La manipulation interactive, à des fins de validation, des mécanismes ainsi construit.
- La validation du choix des composants en termes de consommation électrique ou de poids.
- La gestion de la documentation multimédia en ligne associée aux composants.

Etant donnée la dispersion géographique des équipes impliquées dans les projets du CNES ainsi que la difficulté de communication entre les différents corps de métier, nous avons décidé de proposer une solution basée sur une architecture distribuée afin de pouvoir supporter le prototypage virtuel coopératif synchrone, la maquette virtuelle servant alors de support de communication. L'idée sous-jacente à cette proposition était de fournir, dans le même temps, un système qui puisse servir de support aux processus d'ingénierie concourante adoptés par de nombreux industriels.

Nous allons décrire, dans les chapitre suivant, l'architecture de la plate-forme logicielle VIPER, en charge de la distribution des données et des mécanismes pour le travail coopératif. Nous présenterons ensuite comment cette plate-forme a été utilisée pour développer le système PROVIS.

4 Le système VIPER

Nous avons présenté, dans le chapitre précédent, les spécifications du système PROVIS telles qu'elles ont été définies afin de répondre, du mieux possible, aux besoins exprimés par les utilisateurs finaux que sont les ingénieurs du département mini-satellites du CNES.

Parmi les fonctionnalités identifiées lors de l'étude des besoins, la possibilité de travailler à plusieurs, simultanément et depuis des sites géographiquement distants, nous ont conduits à choisir, comme base de travail, une architecture distribuée supportant le travail coopératif.

C'est pourquoi nous avons utilisé le système VIPER (Virtuality Programming EnviRonment) [Torguet98] pour la définition et la réalisation de nos travaux. VIPER est une plate-forme générique, conforme au paradigme objet, permettant le développement d'applications de réalité virtuelle distribuée. Sa conception et son développement furent menés, à l'Institut de Recherche en Informatique de Toulouse, par Patrice Torguet [Torguet98] en collaboration avec l'auteur du présent mémoire.

Nous présenterons dans ce chapitre la structure logique qui permet de concevoir une application de réalité virtuelle avec VIPER. Nous poursuivrons en détaillant les solutions techniques mises en place pour la réalisation de ce système. Nous présenterons ensuite le modèle que nous proposons pour définir le comportement des objets virtuels ainsi que leurs interactions. Enfin, nous terminerons ce chapitre en présentant les mécanismes que nous avons introduits pour le développement de fonctionnalités de télécommunication entre utilisateurs.

4.1 Introduction

Un système de réalité virtuelle distribuée (RVD) est une architecture matérielle et logicielle répartie sur une ou plusieurs machines et gérant un univers hybride composé de données synthétiques, générées par la machine, auxquelles on ajoute des données en provenance du monde réel. Ces dernières sont fournies au système à l'aide de périphériques spécialisés.

La propriété principale d'un tel système est sa capacité à gérer plusieurs utilisateurs qui, simultanément, partagent le même environnement virtuel tridimensionnel, ces utilisateurs pouvant être séparés, les uns des autres, par des distances très importantes. Le volume des données nécessaires à la représentation d'un tel environnement peut être considérable. De plus, les simulations interactives ont de fortes contraintes temps réel, imposant un délai de réaction du système aussi court que possible (inférieur au $10^{ème}$ de seconde).

Comment alors transférer, à tous les utilisateurs, la quantité de données nécessaires au bon fonctionnement du système, dans un délai extrêmement court, sur des distances pouvant être importantes et ce, tout en respectant la cohérence de la simulation ?

C'est pour répondre à ce problème complexe que plusieurs projets de recherches ont porté, dès le début des années 90 [Blanchard90], sur la mise au point de systèmes de réalité virtuelle distribuée. Les premières solutions proposées avaient comme objectif premier de prouver la pertinence et l'utilité du concept. Plusieurs systèmes de RVD ont depuis vu le jour dans des domaines aussi variés que ceux des simulations militaires [Calvin93] ou des jeux en réseau.

Il existe deux grandes classes de systèmes de réalité virtuelle distribuée. Le lecteur intéressé pourra trouver une étude comparative de ces systèmes dans [Torguet98].

4.1.1 Les systèmes dédiés

De nombreux systèmes de RVD sont dédiés à un certain type d'application comme, par exemple, NPSNET [Macedonia95] pour les simulations militaires ou la machine parallèle de l'IPA Stuttgart [Strommer93] pour le contrôle et la programmation de robots. De tels systèmes tirent profit de leur spécialisation en proposant des techniques d'optimisation propres à leur domaine. Ils sont cependant difficilement utilisables pour d'autres types d'applications. Par exemple, NPSNET n'est réellement efficace que pour la gestion de véhicules terrestres dont les changements de comportements (accélération, décélération, changement de direction) sont peu fréquents. L'introduction d'êtres humains synthétiques [Pratt94] dans ce système entraîne, en autre, une surcharge de communication allant jusqu'à le rendre inutilisable.

4.1.2 Les systèmes génériques

Ces systèmes sont dédiés au développement de n'importe quel type d'applications de RVD. Nous pouvons citer, par exemple, AVIARY [West92], DIVE [Carlsson93] et VLNET [Pandzic95]. La généricité de telles solutions est cependant généralement assurée au détriment des performances de l'application développée.

Des systèmes comme WAVES [Kazman93] ou, plus récemment, World2World [Sense8-98] intègrent, dans leur noyau, les techniques d'optimisations nécessaires au bon fonctionnement du système distribué (dead-reckoning, filtrage de messages, zones de communication, etc.). Nous pensons cependant que ces optimisations devraient être proposées au développeur sous forme de modules. Il serait alors possible d'utiliser, en fonction du type de l'application développée, la technique d'optimisation appropriée, voire même d'en définir une nouvelle afin de traiter un cas particulier.

C'est une des raisons pour lesquelles nous avons conçu le système VIPER [Torguet95] [Balet96] qui propose au développeur un panel de schémas d'optimisation. Comme nous allons le voir dans le paragraphe suivant, son architecture extrêmement modulaire permet, de plus, la création et l'intégration de nouveaux schémas, basés ou non sur ceux disponibles par défaut. Contrairement aux systèmes de RVD évoqués précédemment, le modèle que nous proposons dans VIPER permet de séparer les notions de comportement des notions d'interaction. Le développeur d'applications distribuées a donc la possibilité d'optimiser chacun d'eux séparément, assurant ainsi un réglage très précis des performances obtenues.

4.2 Le système VIPER

Grâce à son architecture répartie, le système VIPER permet de gérer, en temps réel, l'activité simultanée de plusieurs utilisateurs munis de périphériques spécialisés. Cette architecture est générique vis à vis des applications qu'elle peut supporter ainsi que du matériel utilisé. De plus, la principale originalité de cette solution réside dans les deux niveaux de programmation qu'elle offre au développeur.

Le premier niveau masque totalement l'aspect réparti de l'application développée qui apparaît alors comme étant séquentielle. Le second niveau permet de choisir et/ou de redéfinir les mécanismes de répartition de données proposés afin de régler et d'optimiser la distribution en fonction du type d'application développé.

4.2.1 Définition de l'environnement virtuel

La définition d'un environnement virtuel est essentiellement basée sur deux concepts [**Erreur ! Source du renvoi introuvable.**].

Le concept d'entité permet de gérer de façon uniforme l'ensemble des composants qui constituent l'environnement. Il peut s'agir ici d'objets ayant une représentation graphique visible aussi bien que d'objets invisibles définissant des services de l'application développée. Parmi les objets pouvant avoir une représentation graphique, on peut citer les éléments de décors de l'environnement, les éléments ayant une activité autonome ainsi que les clones [Mouli93] qui sont les objets représentant les utilisateurs de l'application. La notion de clone, ou avatar[1] dans la littérature, permet de modéliser de manière homogène le plongement d'un utilisateur dans un environnement de travail tridimensionnel.

Le concept de stimulus permet de modéliser les échanges de données entre entités. Les stimuli sont donc des vecteurs d'informations circulant de manière transparente au sein de l'environnement virtuel. Il existe d'ailleurs plusieurs types de stimuli correspondant chacun à un type d'information échangeable (son, image, ordre, forme, etc.).

[1] Le mot Avatar vient du sanskrit *avatarã* (littéralement descente de Vishnu sur Terre) qui est le nom donné à chacune des réincarnations de Vishnu dans la religion hindoue.

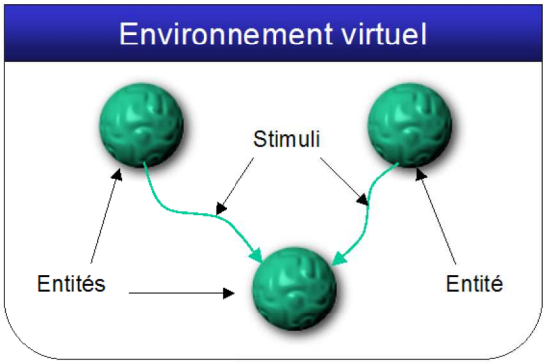


Figure 56 : *Structure d'un environnement virtuel*

4.2.2 Définition d'une entité

Le concept d'entité a été introduit afin de pouvoir modéliser de manière générique l'ensemble des composants de l'environnement virtuel.

Une entité est définie par :

- un ensemble d'attributs dont les valeurs définissent son état interne,
- un comportement définissant son activité propre,
- un ensemble de capteurs servant à recevoir différents types de stimuli,
- un ensemble d'effecteurs permettant d'émettre différents types de stimuli.

Les stimuli destinés à une entité sont recueillis à l'aide de ses capteurs appropriés et peuvent ainsi influencer son comportement. Ce dernier modifie l'état interne de l'entité qui peut, de plus, émettre de nouveaux stimuli, au travers de ses effecteurs, afin de communiquer avec d'autres entités. C'est Whilems [Whilems90] qui a introduit le premier les notions de capteurs et d'effecteurs, concepts fondamentaux de la simulation comportementale formalisant les relations entre l'acquisition des données, leur traitement et la réalisation d'une action.

Dans le cas où une entité avatar représente un utilisateur local du système, un certain nombre de ses capteurs sont directement connectés aux périphériques spécialisés utilisés pour interfacer l'homme avec la machine.

4.2.3 Communications inter-entités

Comme nous l'avons vu, les interactions entre entités sont véhiculées à l'aide de stimuli. Ceux-ci possèdent une étiquette temporelle *Date* qui permet de déterminer leur date d'émission et d'en déduire leur temps de propagation au sein du système. Une liste d'attributs vient compléter la définition d'un stimulus (*Provenance, Destination, Durée de vie, etc.*). De plus, il est possible d'associer, à chacun des attributs et lorsque cela a un sens, une valeur correspondant à la dérivée première par rapport au temps de la valeur de l'attribut, permettant ainsi d'extrapoler sa valeur lorsque la date d'émission n'est pas suffisamment récente.

La circulation des stimuli au sein de l'environnement virtuel distribué est assurée par un mécanisme, appelé espace de stimuli [Figure 57], représentant pour un type de stimulus donné, le milieu d'échange des informations. On peut définir un tel espace comme étant la projection de l'environnement virtuel selon un axe propre à chaque type de stimulus (son, ordre, etc.).

Soit S, l'ensemble de tous les stimuli d'un certain type. L'espace des stimuli associé est alors définit par : $Es = \{s / s \in S\}$

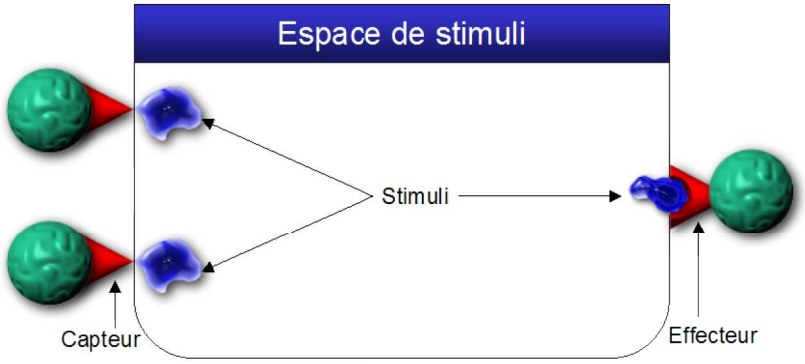


Figure 57 : *Mécanisme de communication entre entités*

4.2.4 Définition d'un capteur

Un capteur de type T a pour rôle de collecter, à une fréquence donnée, sur demande ou automatiquement les stimuli présents dans l'espace de stimuli de type T. Un capteur est composé [Figure 58] d'un module de filtrage, d'une horloge interne synchronisée sur l'horloge de référence du système, d'un module d'interprétation ainsi que d'un tampon mémoire pouvant stocker de 1 à n informations.

Tout stimuli collecté est tout d'abord filtré afin de ne conserver que les informations pertinentes pour l'entité (on rejette les informations trop anciennes ou non destinées à l'entité et on extrapole, si nécessaire, les valeurs pertinentes à partir de leur dérivée première). Une fois cette opération réalisée, les données préservées sont ensuite traduites sous la forme de messages normalisés exploitables par l'entité. Ceux-ci sont stockés dans le tampon mémoire du capteur par ordre de réception. Cette opération peut être automatisée en étant asservie à l'horloge interne du capteur. Il est cependant possible pour une entité, et à tout instant, d'obtenir la valeur courante d'un message et ce, même si de nombreux messages sont déjà disponibles dans le tampon mémoire.

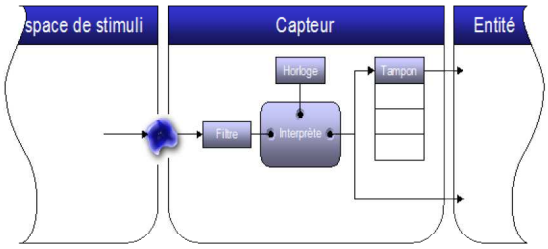


Figure 58 : *Structure d'un capteur*

Il est important de noter qu'assurant la traduction des stimuli, véritables éléments de communication, le module d'interprétation peut permettre à VIPER d'interopérer avec d'autres systèmes répartis.

4.2.5 Définition d'un effecteur

Un effecteur de type *T* a pour rôle d'émettre, sur demande ou à fréquence donnée, des stimuli de ce type vers l'espace de stimuli correspondant. Un effecteur est composé [Figure 59] d'un tampon mémoire pouvant stocker de 1 à n informations, d'un module d'interprétation, d'une horloge interne synchronisée sur l'horloge de référence du système, ainsi que d'un module de filtrage.

Les messages à émettre sont placés par l'entité dans le tampon mémoire d'un de ses effecteurs. Ceux-ci sont alors traduits du mode normalisé, adapté à l'entité, vers le format de stimuli conforme au modèle VIPER. Le module de filtrage permet quant à lui de limiter l'émission de stimuli dans l'environnement virtuel afin, par exemple, d'adapter le débit d'émission à l'engorgement du réseau. Ce mécanisme de communication peut être synchronisé en étant asservi à l'horloge interne de l'effecteur afin, par exemple, d'adapter la fréquence d'émission aux limites du réseau utilisé. De plus, l'entité a la possibilité d'envoyer un message en mode immédiat sans avoir à le placer dans le tampon mémoire d'envoi.

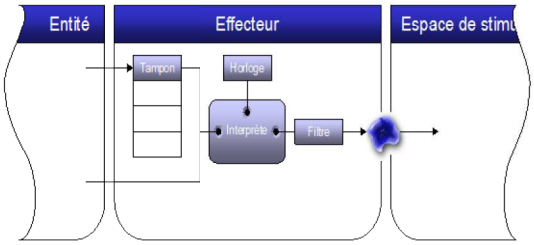


Figure 59 : *Structure d'un effecteur*

4.2.6 Définition de comportements

Chaque entité possède un comportement [Balet96] plus ou moins complexe qui, de plus, peut être modifié dynamiquement au cours de la session de travail. Ce comportement est défini par un ensemble de composants comportementaux interconnectés qui ont la possibilité d'émettre et de recevoir des informations au travers de messages internes normalisés. Ces derniers peuvent être destinés à d'autres composants comportementaux ou à un effecteur qui sera en charge de les transformer sous forme de stimuli puis de les transmettre. Ces composants peuvent de plus réagir aux messages en provenance des capteurs ou d'autres composants ainsi qu'à l'état interne de l'entité.

Un tel composant dispose d'une mémoire propre, d'un ensemble d'attributs définissant son état interne ainsi que d'un mécanisme de fonctionnement pouvant être réalisé suivant différentes manières (langage procédural, automate à états finis [Carlsson93], moteur d'inférence PROLOG associé à une base de règles [Mouli93], etc.).

4.2.6.1 Modèle proposé

Nous proposons d'utiliser le formalisme des réseaux de Petri à objets pour modéliser le comportements des entités [Balet94]. Les réseaux de Petri sont généralement utilisés [Peterson81] pour modéliser le comportement de systèmes parallèles ou temps réel, pour étudier des protocoles de communication ou encore pour le prototypage d'applications interactives. Ils sont donc parfaitement adaptés à notre problème.

Présentés pour la première fois en 1962 par C. A. Petri [Petri62], ces réseaux ont connus depuis de nombreuses extensions comme les réseaux de Petri de haut niveau présentés par H. J. Genrich [Genrich81], les réseaux de Petri colorés [Jensen92] ou encore les réseaux de Petri à objets [Sibertin85].

Nous utiliserons ces derniers pour leur simplicité d'intégration, leur complémentarité et leur cohérence vis à vis du paradigme objet retenu pour nos travaux [Meyer88].

Les réseaux de Petri à objets introduisent une nouvelle dimension au paradigme objet. Ils permettent en effet de définir des comportements décrivant l'activité spontanée de l'objet, l'effet de son état interne sur les opérations qu'il offre et, réciproquement, l'effet de ces dernières sur son état interne. Ils possèdent en outre l'avantage d'être basés sur des concepts mathématiques permettant la vérification statistique de propriétés importantes comme le déterminisme ou le non blocage du comportement en cours d'exécution.

4.2.6.2 Réseaux de Petri marqués

Un réseau de Petri marqué est défini par le quintuplet $R = <P, T, Pre, Post, M>$ où :

P est un ensemble fini de places, une place représentant un état déterminé du système.

T est un ensemble fini de transitions. On définit une transition comme étant un opérateur de changement d'état.

Pre est l'application incidence avant, $P \times T \rightarrow N$, déterminant les places précédentes.

Post est l'application incidence arrière, $P \times T \rightarrow N$, déterminant les places suivantes.

M est l'application $M : P \rightarrow N$ déterminant le marquage initial du réseau, à savoir le nombre de marques (plus communément appelés jetons) pour chacune des places de l'ensemble *P*. *M(p)* représente donc le nombre de jetons contenus dans la place *p*.

L'état du système est alors déterminé par la position des jetons parmi les places du réseau. Les places et les transitions sont interconnectées à l'aide d'arcs orientés définissant ainsi les possibilité d'évolution des jetons au sein du réseau.

4.2.6.3 Graphe associé et notation matricielle

On peut associer, à tout réseau de Petri, une représentation symbolique sous forme de graphe [Figure 60] possédant deux types de nœuds : les places, symbolisés par des cercles, et les transitions, symbolisées par des boites rectangulaires. Un arc relie alors une place p à une transition t si et seulement si $Pre(p,t) \neq 0$. De manière similaire, un arc relie une transition t à une place p si et seulement si $Post(p,t) \neq 0$. On étiquette de plus les arcs du graphe avec les valeurs non nulles des matrices *Pre* et *Pos*.

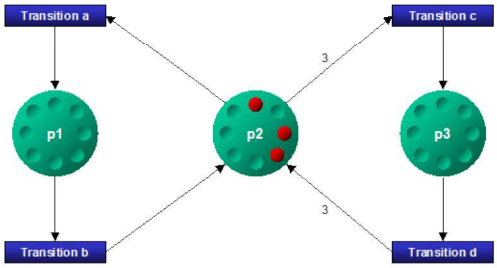


Figure 60 : *Exemple de réseau de Petri marqué*

Le marquage M peut être représenté par un vecteur ayant pour dimension le nombre de places, soit $card(P)$; Pre, $Post$ seront alors des matrices dont le nombre de lignes est égal à $card(P)$ et le nombre de colonne est égal à $card(C)$. On note $Pre(.,t)$ et $Post(.,t)$ les colonnes de ces matrices associées à la transition t ; on obtient alors $dim(Pre(.,t)) = dim(Post(.,t)) = card(P)$.

Le réseau présenté Figure 60 est alors défini comme suit :

- $P = \{p1, p2, p3\}$,
- $T = \{a, b, c, d\}$,
- $$Pre = \begin{matrix} a & b & c & d \end{matrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} p1 \\ p2 \\ p3 \end{matrix}$$
- $$Post = \begin{matrix} a & b & c & d \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{matrix} p1 \\ p2 \\ p3 \end{matrix}$$
- $$M = \begin{bmatrix} 0 \\ 3 \\ 0 \end{bmatrix} \begin{matrix} p1 \\ p2 \\ p3 \end{matrix}$$

Il est possible d'utiliser la représentation matricielle d'un réseau de Petri pour vérifier mathématiquement un certain nombre de ses propriétés structurelles [Lautenbach87]. On dit, par exemple, qu'un réseau de Petri R est pur lorsqu'il ne comprend aucune boucle élémentaire, c'est à dire aucune transition n'ayant la même place en entrée et en sortie. Cette propriété est vérifiée si et seulement si $\forall p \in P$ et $\forall t \in T : Pre(p,t).Post(p,t) = 0$. De même, il est possible de vérifier, par calcul matriciel, l'existence de différents types de conflits (structurels, effectifs) ou l'accessibilité des places au sein du réseau.

4.2.6.4 Interface réseau/environnement

La modélisation de comportements réactifs à l'aide de réseaux de Petri nécessite la représentation de l'interface entre le réseau et son environnement. Deux approches sont alors possibles afin de répondre à ce problème.

Ou bien l'on considère un sous-ensemble de T comme étant l'ensemble des transitions de type interface déclenchées par l'environnement, ou bien l'on considère que ce dernier peut déposer des jetons dans certaines places de type interface. La première solution n'est pas convenable dans notre cas car elle implique que l'environnement déclenche directement certaines actions du système. La seconde approche permet, quant à elle, le déclenchement de plusieurs actions dès la réception d'un événement ; elle est ainsi plus cohérente vis à vis du modèle. En effet, il suffit pour cela de représenter la réception d'un événement extérieur, un message dans notre système, par le dépôt d'un jeton dans une place.

4.2.6.5 Réseaux de Petri à objets

Depuis leur introduction, plusieurs extensions du modèle de réseau de Petri traditionnel ont donc été proposées afin d'accroître leur capacité de modélisation et de prendre ainsi en compte les évolutions majeures qui ont marqué le domaine du génie logiciel. En effet, les réseaux de Petri ne différencient pas, dans leur formalisme premier, les différents types de jetons pouvant circuler sur le réseau. Au contraire, ils se concentrent essentiellement sur la gestion des structures de contrôle sans tenir compte des structures de données utilisées. De plus, ils ne permettent pas de structurer un modèle de système sous une forme hiérarchique qui leur permettrait de limiter leur taille, restant ainsi représentable graphiquement.

Plusieurs extensions ont donc été proposées parmi lesquelles nous avons choisi celle des réseaux de Petri à objets (RPO) [Bastide89].

Les RPOs [Figure 61] ont pour caractéristique première de permettre l'étiquetage des arcs à l'aide d'objets qui sont utilisés comme paramètres formels des transitions, définissant ainsi les flux d'objets au travers du réseau. De plus, la définition du jeton est étendue afin de pouvoir être instance d'une classe quelconque. Les transitions, quant à elles, sont dotées d'une partie préconditionnelle déterminant si leur activation est possible. Dans le cas où cette dernière aurait lieu, un jeton est retiré de sa place d'origine, traité par la transition puis retransmis vers une place destination ou bien détruit.

La retransmission d'un jeton peut être orientée vers une place donnée en fonction de l'état interne de la transition. Pour ce faire, une transition dispose d'un ensemble de règles d'émission associées aux arcs de sortie.

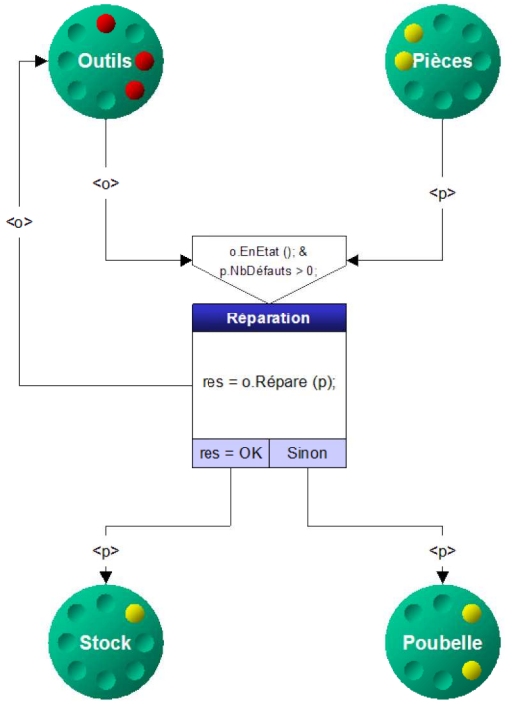


Figure 61 : *Modélisation d'un système de réparation de pièces par réseau de Petri à objets (o est un outil et p une pièce)*

4.2.6.6 Mise en place du modèle

Le comportement d'une entité est, nous l'avons vu, constitué d'un ensemble de composants comportementaux interconnectés. Nous proposons donc de modéliser un comportement à l'aide d'un RPO où les transitions représenteraient les composants comportementaux. Chacun d'eux est disponible sous la forme d'une librairie dynamique (dll ou dso) chargeable et modifiable au cours du déroulement de la simulation.

Prenons pour exemple une entité composteur de billets. Celle-ci est équipée d'un capteur de collisions qui détecte la présence d'un billet sous le poinçon. Elle dispose, de plus, d'un effecteur de sons lui permettant d'émettre un signal si le billet (<b>) est soit mal positionné, soit déjà poinçonné. Le comportement [Figure 62] de cette entité est défini par deux modules comportementaux. Le premier est en charge de faire un trou dans la forme d'une entité billet si le poinçon est en état de fonctionner. Le deuxième se charge d'émettre un signal sonore, si l'alarme fonctionne, dans le cas où le billet ne conviendrait pas.

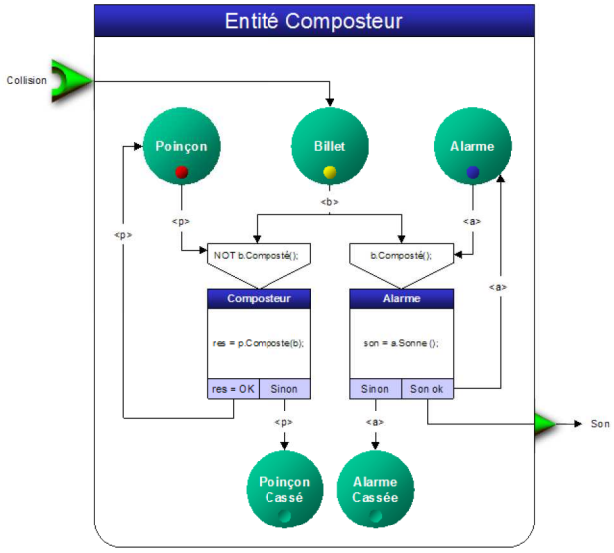


Figure 62 : *Comportement de l'entité composteur*

Lorsque le capteur détecte la présence d'un billet sous le poinçon, il dépose, conceptuellement parlant, un jeton billet dans la place correspondante. Cette action a pour effet de rendre possible l'exécution de transitions et de mettre ainsi en marche le comportement. Cet exemple a été implanté dans une application [David97] développée par CISI pour la SNCF.

4.2.7 Niveaux de détails des comportements

Il arrive parfois que les comportements définis soient lourds en temps de calcul, pour un résultat pas toujours visible. C'est le cas, par exemple, pour une entité personnage dont le comportement assure l'animation de ses différents membres par déformation de maillage. Ces calculs pourraient être grandement simplifiés, sans pour autant perdre en qualité d'animation, lorsque le personnage se trouve éloigné de l'utilisateur. Il suffirait dans ce cas de ne plus déformer le maillage mais de se contenter de modifier l'orientation des membres virtuels.

C'est pour traiter ce type de cas que VIPER propose un mécanisme de niveaux de détails de comportements. Le principe consiste à choisir le comportement d'une entité en fonction de sa distance au point de vue. En fait, nous utilisons la taille de sa boite englobante, projetée sur l'écran, comme critère de sélection du comportement. Ce mécanisme est bien plus pertinent que la simple utilisation d'une distance puisqu'il permet de prendre en compte la taille de l'entité.

Le mécanisme de changement de niveau de comportement est délicat à mettre en place. Il faut en effet s'assurer que le comportement utilisé est dans un état stable, c'est à dire qu'aucune procédure n'est en cours d'exécution. La modélisation comportementale par RPO facilite grandement cette tâche puisqu'il suffit de s'assurer, pour cela, qu'aucun jeton ne circule dans le réseau.

4.3 Gestion des collisions

La détection des collisions entre les différentes entités qui composent un environnement virtuel est fondamentale. C'est elle qui permet de garantir l'intégrité de la scène en évitant que deux entités rigides ne s'interpénètrent. Elle constitue donc l'une des bases des techniques de l'interaction homme machine 3D puisqu'elle pourra, ou non, autoriser certaines manipulations d'objets en fonction des contacts générés.

VIPER propose de gérer la détection des contacts à l'aide de capteurs de collisions. Une entité disposant d'un tel capteur est automatiquement enregistrée par le système comme étant sensible aux collisions. Le système détermine, à chaque pas de simulation, si une telle entité interpénètre un autre élément de l'environnement. Si tel est le cas, l'entité reçoit un stimuli de collision l'en informant. C'est ensuite à l'entité de décider, en fonction du comportement qui lui est attribué, si elle doit retourner à sa position antérieure, rebondir ou subir une déformation..

La détection d'un contact entre deux entités est assurée à l'aide de la bibliothèque V-COLLIDE [Hudson97], basée sur la méthode des arbres de boites englobantes orientées [Gottshalk96]. Cette bibliothèque offre le double avantage d'être très performante et d'être capable de traiter, sans contraintes de topologie, n'importe quel type de géométries. Le résultat fourni par l'algorithme, appliqué à deux entités en contact, est de la forme :

$$Res = \left\{\left(f_1, f_1^{'}\right), \ldots, \left(f_n, f_n^{'}\right)\right\} \text{ où } \left(f_i, f_i^{'}\right) \text{ est un couple de facettes en contact.}$$

Compte tenu du haut niveau d'interactivité que nous souhaitons offrir à notre système de prototypage virtuel, nous avons intégré, dans VIPER, un mécanisme d'optimisation, nommé cache de collisions, permettant de limiter le nombre de traitements nécessaires à la gestion des collisions. Comme nous l'avons vu précédemment, une entité en contact détermine elle-même le comportement à adopter. Soit elle revient à une position antérieure, soit, par exemple, elle reste en place et émet un signal sonore. Sans optimisation, cette deuxième situation engendrerait un grand nombre de tests inutiles puisque le système détecterait une collision et calculerait les facettes en contact à chaque pas de simulation.

Nous proposons donc d'enregistrer dans un cache les couples d'entités qui s'interpénètrent, le résultat de l'algorithme V_COLLIDE ainsi que la date du calcul. On obtient donc, pour chaque collision détectée le n-uplet suivant :

$$\{\text{date, @entité1, @entité2, } \left\{\left(f_1, f_1^{'}\right), \ldots, \left(f_n, f_n^{'}\right)\right\}\}$$

Nous associons, de plus, à chaque attribut position, orientation et échelle d'une entité, une étiquette temporelle déterminant la date de dernière mise à jour de l'attribut :

Position :
    Valeur
    Date
Orientation :
    Valeur
    Date
Echelle :
    Valeur
    Date

L'algorithme de gestion des collisions est alors le suivant :

```
Pour chaque entité E1 disposant d'un capteur collision
{ Pour chaque entité E2 exceptée E1
    { C = RechercheCollisionDansCache {E1, E2};
      Si
        C != NULL et
        (E1.Position.Date <= C.d et E1.Orientation.Date <= C.d et E1.Echelle.Date <= C.d) et
        (E2.Position.Date <= C.d et E2.Orientation.Date <= C.d et E2.Echelle.Date <= C.d)
      Alors
        Stocker C.Res;
      Sinon
        Resultat = CalculCollision (E1, E2);
        SupprimeDansCache (C.Res);
        PlaceDansCache {date_courante, E1, E2, Resultat};
        Stocker Résultat;
      FinSi
    }
}
```

La mise en place et la distribution du cache de collisions permet aussi d'optimiser le nombre de traitements dans le cas de sessions impliquants plusieurs utilisateurs. En effet, plutôt que de calculer sur chaque machine l'état des collisions, le système d'un utilisateur peut retrouver des informations de collisions placées dans le cache à la suite d'une manipulation opérée par un autre utilisateur.

4.4 Téléconférence

Lors d'une session de travail impliquant plusieurs utilisateurs situés à des endroits géographiquement éloignés, le simple partage de l'environnement virtuel ne suffit pas à donner une dimension collective satisfaisante aux tâches effectuées. En effet, les utilisateurs ont besoin de pouvoir se parler afin d'échanger idées et points de vue, la parole constituant le mode de communication le plus simple et le plus naturel pour l'homme.

Pour ces raisons, nous avons étudié puis mis en place une solution permettant de gérer, de manière optimisée, les flux audio et vidéo au sein du système VIPER. Ainsi, un utilisateur peut voir, être vu et dialoguer avec des utilisateurs partageant le même environnement de travail.

Les solutions envisagées ont été choisies parmi les standards en vigueur afin de permettre l'interopérabilité de notre système avec des outils de visioconférence extérieurs. Chacune de ces solutions a été évaluée en termes de qualité et de performances (débit généré, temps de calcul) afin de retenir celles offrant le meilleur compromis.

4.4.1 Connexion audio

Chaque utilisateur de notre système a la possibilité de transmettre et de recevoir des flux audio afin de communiquer oralement avec les autres utilisateurs du système. Nous avons, pour cela, mis en place un espace de flux audio assurant la distribution rapide des stimuli audio au sein de l'environnement virtuel.

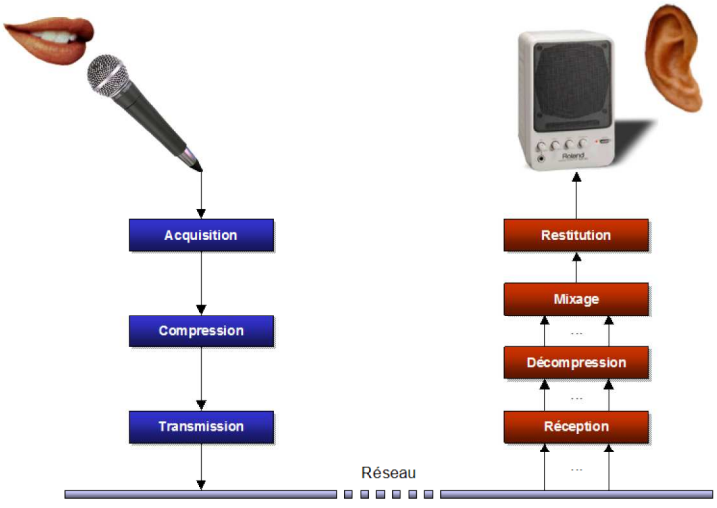


Figure 63 : *Transmission d'un flux audio sur un réseau informatique*

Nous présentons [Figure 63] les différentes phases du processus de transmission d'un signal audio entre deux utilisateurs distants.

Le processus séparant l'acquisition de la restitution doit être réalisé en un temps inférieur à la durée du signal audio en entrée afin de conserver la continuité du flux traité ; hacher un flux audio génère systématiquement un inconfort d'écoute sérieux pour l'utilisateur.

Une connexion audio doit de plus être au minimum bidirectionnelle afin d'autoriser le dialogue simultané entre deux utilisateurs.

4.4.1.1 L'acquisition audio

C'est lors de cette phase que le signal vocal ou musical, en général analogique, est transformé en un signal numérique. Le mécanisme mis en œuvre, appelé échantillonnage [Figure 64], consiste à mesurer la valeur courante du signal analogique et à la convertir en une valeur numérique. La fréquence et la précision de la mesure constituent les deux facteurs caractérisant la qualité du processus.

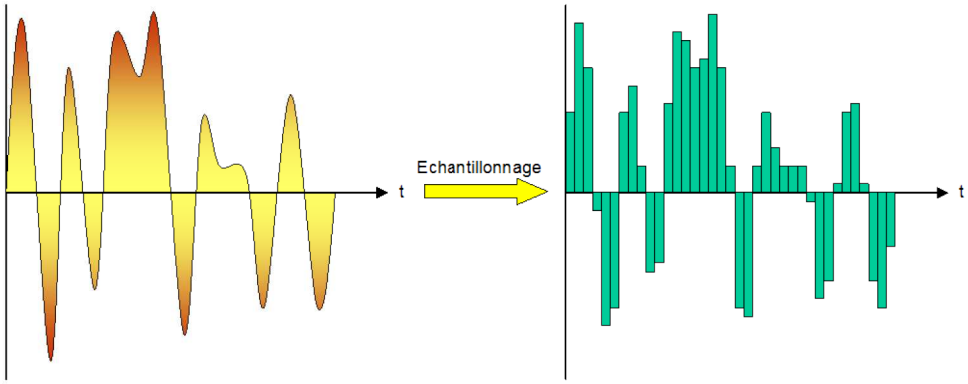


Figure 64 : *Echantillonnage d'un signal analogique*

Le théorème de l'échantillonnage de Nyquist établit qu'un signal analogique peut être reconstruit à partir des échantillons numérisés si la fréquence d'échantillonnage est au moins deux fois supérieure à la bande passante du signal original. Les caractéristiques psychoacoustiques de l'audition humaine sont telles que seules les fréquences entre 20 Hz et 20 KHz peuvent être perçues.

Chaque échantillon est codé sur un nombre donné de bits. Plus ce nombre est grand et mieux la dynamique et la définition du son original sont conservées. Par exemple, la qualité Compact Disc correspond à une fréquence d'échantillonnage de 44,1 KHz, donc supérieure à 2 * 20 KHz, pour une précision de 16 bits.

4.4.1.2 La compression audio

Une seconde de musique numérisée au format Compact Disc, en monophonie, correspond à 44100*16 ≈ 705 Kbits. Pour référence, un réseau local traditionnel (Ethernet) peut supporter théoriquement 10 Mbits ou 100 Mbits par seconde. Ces performances ne sont jamais atteintes car, généralement, plusieurs stations de travail partagent le même réseau. De plus, de nombreuses données transitent sur celui-ci afin d'en assurer le bon fonctionnement, diminuant encore la bande passante disponible. Il n'est donc pas envisageable de transférer le résultat de l'échantillonnage tel quel sans risquer de congestionner le réseau. La situation est encore plus critique lorsqu'il s'agit de gérer le dialogue entre plusieurs utilisateurs, d'emprunter un réseau à grande échelle (WAN) ou de permettre la participation d'un utilisateur connecté à l'aide d'une ligne RNIS (Numéris, 64 Kbits par seconde).

C'est pourquoi les données échantillonnées doivent être compressées afin de pouvoir occuper un minimum de place. Pour ce faire, deux mécanismes sont utilisés. Le premier consiste à prendre en compte les caractéristiques psychoacoustiques de l'audition chez l'être humain afin de restreindre la bande passante des fréquences traitées. La Figure 65 présente l'audiogramme inverse moyen obtenu en plaçant une personne dans une pièce isolée et en lui demandant d'indiquer, pour chaque fréquence émise, le seuil à partir duquel elle discerne un son. On montre ainsi que l'oreille humaine est particulièrement sensible aux fréquences entre 2 et 4 KHz.

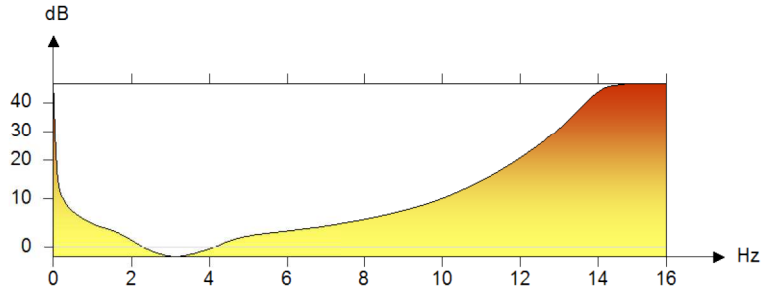


Figure 65 : *Seuil de silence*

On peut donc limiter, en accord avec le théorème de Nyquist, la fréquence d'échantillonnage à 8 KHz tout en conservant une représentation correcte du signal. Cette valeur possède de plus l'avantage d'être adaptée à la bande passante (3,5 KHz) des paires torsadées équipant les réseaux téléphoniques. C'est d'ailleurs cette méthode, appelée PCM (Pulse Coded Modulation), qui est utilisée pour la communication téléphonique traditionnelle. L'échantillonnage est réalisé avec un échelle logarithmique sur 8 bits, ce qui équivaut à un échantillonnage sur 13 bits avec une échelle linéaire. Ainsi, une seconde de son numérisé en PCM occupe 64 Kbits, ce qui est encore trop pour une conversation impliquant plusieurs utilisateurs.

Le deuxième mécanisme consiste à utiliser un algorithme de compression, appelé codeur, afin de réduire encore la quantité de données nécessaires au transport d'un flux audio. Les méthodes de compression traditionnelles, sans perte d'informations (Huffman, LZW, etc.), ne donnent généralement pas de bon résultats [Sayood96] pour la compression de données audio. C'est pourquoi plusieurs méthodes ont été développées pour répondre spécifiquement à ce problème. Celles-ci ont en commun la caractéristique de compresser les données avec pertes d'informations, ce qui, pour un signal audio peut être justifié. On montre par exemple, à l'aide d'audiogrammes, que l'émission d'un son bref (5 ms) immédiatement suivi de celle d'un son fort (60 dB) masque la perception des sons plus faibles (40 dB) suivants pendant environ 50 ms.

Codage différentiel

Le codage différentiel constitue un mécanisme de compression basé sur l'observation selon laquelle les échantillons successifs d'une source audio sont fortement corrélés. Il semble donc judicieux de ne plus coder les échantillons eux-mêmes mais plutôt la différence entre deux échantillons successifs.

En pratique, la corrélation des échantillons permet de prédire l'échantillon $x_{(n+1)}$ à partir des échantillons x_n, $x_{(n-1)}$, $x_{(n-2)}$, etc. Ainsi, plutôt que de coder un échantillon x_n, on codera la différence entre celui-ci et sa prédiction. C'est sur ce principe que fonctionnent les algorithmes de type DPCM (Differential PCM) ou ADPCM (Adaptive Differential PCM).

Le codage DPCM utilise $x_{(n-1)}$ comme valeur prédictive de l'échantillon x_n. Il s'agit donc d'une méthode purement différentielle.

Le codage ADPCM [Sherif93] détermine, quant à lui, une valeur de prédiction en fonction des p échantillons précédents :

$$\tilde{x}_n = \sum_{i=1}^{p} a(i) x_{(n-i)} \quad \text{où } \tilde{x}_n \text{ est la valeur de prédiction pour l'échantillon } x_n$$

On détermine les coefficients $a(i)$ en minimisant la variance de l'erreur de prédiction $d(n) = x(n) - \tilde{x}(n)$.

Codage par synthèse

Contrairement aux méthodes différentielles qui traitent directement le signal numérisé, le codage par synthèse consiste à construire, pour un bloc d'échantillons donné, un modèle permettant de générer un bloc d'échantillons statistiquement proches. Il s'agit donc de modéliser le système de production du son, en l'occurrence le système vocal (poumons, cordes vocales, trachée, gorge, bouche, lèvres).

Le codage LPC (Linear Predictive Coding) modélise ce dernier à l'aide d'un ensemble de cylindres de diamètres différents, 10 dans le cas de LPC-10 [Tremain82], excités par un signal sinusoïdal ou un bruit blanc. Le choix de la méthode d'excitation repose sur la nature voisée (« a », « u ») ou non (« r », « s ») du signal traité. Ce codage se décompose donc en deux étapes.

La première consiste à déterminer quelle méthode d'excitation choisir en fonction de la nature du signal d'entrée. Celui-ci subit un filtrage passe bas, l'analyse de l'autocorrélation du signal ainsi obtenu permet alors de déterminer simplement la nature du signal d'entrée.

La deuxième étape consiste, quant à elle, à déterminer le diamètre des cylindres. Pour cela, on utilise l'algorithme de Durbin [Rabiner78] afin de déterminer les coefficients de réflexion du signal. Ces coefficients définissent le diamètre des cylindres.

Ces deux étapes sont exécutées toutes les 20 ms, soit pour chaque bloc de 160 échantillons (pour une fréquence d'échantillonnage de 8 KHz). La méthode génère, en sortie et pour chaque bloc, une fréquence d'excitation codée sur 16 bits, un ensemble de 10 coefficients, codés chacun sur 8 bits, et un gain codé sur 8 bits. On obtient alors 104 bits pour 20 ms, soit un débit de 5,2 Kbits par seconde. Les performances du codage LPC en terme de compression sont très bonnes (> 12 :1). Malheureusement, utilisé tel quel, ce codeur ne permet pas la reconstruction d'un signal audio avec une qualité suffisante.

Le codage CELP (Code Excited Linear Predictive) [Campbell91] améliore grandement la qualité de restitution du codage LPC en introduisant de nouvelles méthodes d'excitation, combinaisons linéaires de fonctions stochastiques. Cependant, le choix de la méthode d'excitation à utiliser en fonction du signal traité est très coûteuse en temps de calcul, imposant par là même l'adjonction d'une carte spécialisée pour le traitement du signal si l'on souhaite conserver de la puissance machine pour d'autres tâches.

Le codage GSM fut proposé par le Groupe Spécial Mobile dans le cadre du développement de l'architecture du réseau de communication Européen pour la téléphonie mobile. Il s'agit d'une variante de la méthode CELP, appelée RPE (Regular Pulse Excited), intégrant une fonction d'excitation faisant intervenir des pulsions périodiques couplées à une boucle de prédiction à long terme. Les détails de la méthode, très complexe, peuvent être trouvés dans [Vary89] [Degener94].

Les différents codeurs audio adaptés au traitement de la parole sont présentés en détails dans [Vega García96] [Sayood96].

4.4.1.3 Performances des différents codeurs

Rappelons que notre objectif est de transférer, au travers d'un réseau informatique, le flux audio que constitue un dialogue oral en utilisant un débit et un temps de traitement minimaux tout en conservant l'intelligibilité du signal transmis. Nous avons implanté les différentes méthodes de codage et de décodage audio, présentées dans ce chapitre, sur un système Windows95 équipé d'un Pentium II cadencé à 266 Mhz. Nous en proposons [Tableau 2] une évaluation en termes de débit généré, de qualité et de temps de traitement pour une trame de 1 seconde, échantillonnée à 44,1 KHz , en 16 bits.

Le critère de qualité, bien que très subjectif, peut être mesuré par le rapport signal/bruit (SNR) du signal reconstruit :

$$SNR = 10\log_{10}\left(\frac{\frac{1}{N}\sum_{i=0}^{N-1} x_i^2}{\frac{1}{N}\sum_{i=0}^{N-1}\left(x_i - \tilde{x}_i\right)^2}\right)$$

où x_i représente le i^{ème} échantillon du signal d'entrée et $\tilde{x}_i$, le i^{ème} échantillon du signal reconstruit, N étant le nombre d'échantillon du signal. Dans le cas d'un signal obtenu par échantillonnage via un micro, le rapport signal/bruit doit aussi intégrer le bruit de quantification généré par la différence entre l'échantillon obtenu et la valeur réelle ; le lecteur pourra se reporter à [Vega García96] pour de plus amples détails. Le SNR ne peut cependant pas prendre en compte l'intelligibilité du signal reconstruit. Il existe pour cela un protocole d'évaluation, appelé Mean Opinion Score (MOS), qui permet de noter une méthode en fonction de tests pratiques (mesure de compréhension d'une phrase, robustesse à l'engorgement du réseau, etc.).

Le choix du type de codage doit donc prendre en compte un critère de qualité, un critère de performance mais il doit aussi tenir compte de l'effort de normalisation conduit par l'International Telecommunication Union (ITU). Cet organisme propose, dans la norme H.32x, plusieurs standards de codage audio (G.7xx) adaptés au type de réseau utilisé. Le respect des recommandations de l'ITU permet d'assurer l'interopérabilité avec d'autres systèmes de téléconférence.

Norme	Méthode	Débit (Kbits/s)	Temps (ms) Compression / Décompression	Qualité (MOS[1])
G.711	PCM (aucune compression)	64	1 / 1	4,0
G.722	Sub-Band ADPCM	64[2]	11/8	3,6
GSM	GSM	13,1	32 / 15	3,7
G.723	ADPCM	40, 32 ou 24[3]	28 / 12	3,8
G.723.1	MP-MLQ CELP	6,4	114 / 21	3,9
	Algebraic CELP	5,3	156 / 23	3,9
G.728	Low Delay CELP	16	125 / 29	4,1

Tableau 2 : *Débit, temps de traitement et qualité des différents codeurs audio normalisés*

Nous avons retenu pour notre système la norme GSM dans sa version 10 (GSM 06.10 RPE-LTP). Elle possède l'avantage d'être peu coûteuse en temps machine et de fournir un excellent taux de compression (5:1) tout en permettant la reconstruction d'un signal de très bonne qualité. Il est à noter que cette méthode était jugée, il a de cela deux ans à peine, trop gourmande en temps de calcul pour être utilisée sur une machine de type PC [Vega García96]. L'apparition de la technologie MMX a offert la possibilité de l'utiliser sans avoir recours à l'adjonction d'une carte spécialisée pour le traitement du signal.

4.4.2 Connexion vidéo

Plusieurs études [Yankee95] ont montré que la communication orale pouvait s'avérer frustrante lorsqu'elle était découplée de toute représentation physique des interlocuteurs. C'est pourquoi VIPER intègre des fonctionnalités pour la gestion de flux vidéo, permettant ainsi à un utilisateur d'entrer en contact vidéo avec d'autres utilisateurs.

La Figure 66 présente les différentes phases du processus de transmission d'un signal vidéo entre deux utilisateurs distants.

[1] Sources AT&T Bell Labs

[2] La bande passante du signal d'entrée est étendue à 7,5 KHz

[3] Les mesures ont été effectuées en mode 24 Kbits par seconde

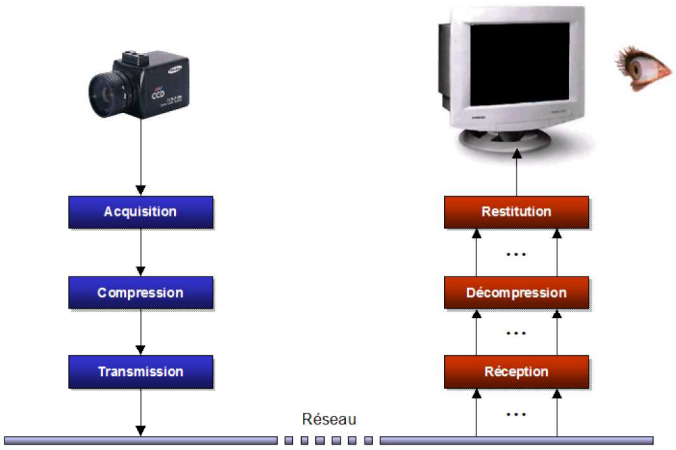


Figure 66 : *Transmission d'un flux vidéo sur un réseau informatique*

Contrairement au flux audio qui ne tolère aucune perte ou retard d'informations sans désagrément pour l'utilisateur, le flux vidéo peut être interrompu ou ralenti sans inconvénient majeur lors d'une session de téléconférence. Il est ainsi possible de l'adapter en fonction de l'engorgement du réseau ou des ressources système disponibles. Un flux cadencé entre 5 et 10 images par seconde est suffisant pour fournir une sensation visuelle satisfaisante pour ce type d'application. Il est aussi envisageable de réduire ponctuellement ce débit aux alentours d'une image par seconde sans générer d'inconfort visuel pour l'utilisateur ; la liaison vidéo étant utilisée pour appuyer le dialogue, elle ne constitue donc pas un vecteur sémantique important dans le cadre d'un collecticiel.

4.4.2.1 L'acquisition vidéo

C'est lors de cette phase que le périphérique d'acquisition, en l'occurrence la caméra, capture les images qui lui sont présentées afin de les transmettre à l'ordinateur. Ce processus, appelé numérisation, fourni en sortie un flux d'images numériques. Chaque élément d'image (pixel) possède une couleur définie par trois composantes de couleur rouge (R), verte (V) et bleue (B).

Un flux vidéo peut être qualitativement quantifié à l'aide de trois paramètres, la résolution des images le composant, la précision de la mesure, ainsi que la fréquence d'acquisition. La résolution détermine le nombre de pixels qu'une image comporte verticalement et horizontalement. La précision de la mesure défini le nombre de bits alloués pour représenter les composantes de couleur d'un pixel, elle défini donc directement le nombre de couleurs disponibles pour représenter l'image (un codage sur 24 bits, soit 8 bits pour chacune des composantes, offre une palette d'environ 16 millions de couleurs). La fréquence d'acquisition détermine le nombre d'images que le système est à même d'acquérir par seconde.

4.4.2.2 La compression vidéo

Le débit généré par le transfert d'un flux vidéo est largement supérieur à celui généré par le transfert d'un flux audio de même durée. Par exemple, le transfert d'un flux vidéo, en 352*288 pixels sur 8 bits et à 10 Hz, nécessite, sans optimisation, un débit de 8 Mbits par seconde.

Il est donc là encore nécessaire de mettre en œuvre des algorithmes de compression afin de réduire considérablement la quantité de données transmises. Il existe pour cela plusieurs méthodes de codage normalisées.

Codage MJPEG

Le codage Motion-JPEG (MJPEG) fut l'un des premiers utilisés pour le transfert de flux vidéo sur Internet. Il est directement basé sur le format de compression avec pertes[1] d'images fixes, JPEG, défini par le comité « Joint Photographic Experts Group » regroupant le CCITT et l'ISO [Wallace91]. Le but de ce groupe de travail était de définir une méthode de compression pour les images photographiques qui soit à la fois à la pointe des techniques de compression et très polyvalente (applicable à tout type d'images et sur tout type de machines). Ces travaux ont débuté en juin 1987 et ont conduit à une norme ISO publiée en 1993.

Le codage MJPEG revient à compresser chaque image du flux vidéo au format JPEG, opération pouvant être décomposée en 4 étapes :

Transformation du mode de représentation de l'image : les composantes RVB sont transformées par matriçage en trois variables, la luminance Y ainsi que les chrominances rouge U et bleue V (U = R – Y et V = B – Y). La luminance représente l'image d'origine en niveaux de gris alors que les chrominances correspondent à la coloration de l'image. On montre que les composantes YUV sont moins corrélés que les composantes RGB et que la plus grande partie des informations est contenue dans Y. De plus, chez l'homme, l'œil est bien moins sensible aux informations de chrominances qu'à celles de luminance. C'est pourquoi il est possible, dès cette étape, d'effectuer une première opération de compression en sous-échantillonnant les deux coefficients de chrominance [CCIR601]. Une composante U, ou V, correspondra alors à 4 pixels de l'image d'origine produisant ainsi une compression d'un facteur 2.

[1] JPEG définit également une compression conservative.

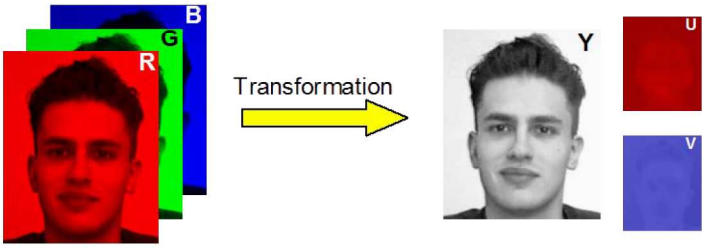


Figure 67 : *Chaque image RGB est transformée en une image au format YUV*

Codage par transformation orthogonale : l'objectif de cette étape est de changer l'espace de représentation de l'image afin d'isoler les zones où se trouve l'énergie du signal, c'est à dire son information utile. Une image est à l'origine représentée dans le domaine spatial ; on applique alors un codage par Transformée en Cosinus Discrète (TCD) [Ahmed74] pour obtenir sa représentation dans le domaine fréquentiel.

Pour ce faire, on découpe tout d'abord l'image en blocs de 8*8 pixels auxquels on applique la transformation TCD définie par :

$$F(u,v) = 1/4C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos(\pi(2x+1)u/16)\cos(\pi(2y+1)v/16)$$

pour u et v variant de 0 à 7 avec $C(0) = \frac{1}{\sqrt{2}}$ et $C(i) = 1$ pour $i \in [1,7]$.

On obtient alors une table de 8*8 valeurs représentant les coefficients fréquentiels $F(u, v)$ de la TCD ; plus leur indice de colonne, respectivement de ligne, est important et plus la fréquence verticale, respectivement horizontale, correspondante est grande [Figure 68]. En pratique, on observe que cette table contient beaucoup de zéros dans son coin inférieur droit, indiquant par là même que les images sont généralement composées de basses fréquences.

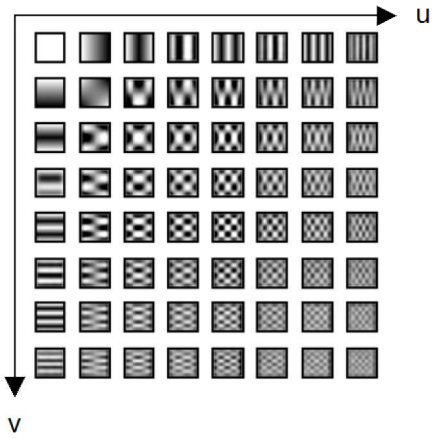


Figure 68 : *Les 64 coefficients fréquentiels de la TCD représentés sous forme graphique*

La transformée inverse, utilisée lors de la phase de décodage, est définie par :

$$f(x,y) = 1/4 \sum_{u=0}^{7} \sum_{v=0}^{7} C(u)C(v)F(u,v)\cos(\pi(2x+1)u/16)\cos(\pi(2y+1)v/16)$$

pour x et y variant de 0 à 7 avec $C(0) = \frac{1}{\sqrt{2}}$ et $C(i) = 1$ pour $i \in [1,7]$.

Quantification : une quantification est alors appliquée, pour chaque table obtenue, afin d'accroître le nombre de coefficients fréquentiels à valeur nulle. C'est ici que la dégradation de l'image intervient car l'opération de quantification inverse ne permettra de retrouver qu'une valeur approchée des coefficients initiaux [Scotton93].

Pour réaliser cette opération, il existe deux types de quantification. La première, appelée quantification uniforme, vise à diviser les coefficients contenus dans la table par un coefficient de quantification pondéré par le facteur de qualité d'image Q choisi, $Q \in [1,100]$. Le résultat de la division est converti sous la forme d'un entier, générant ainsi une perte d'information et donc la compression recherchée. La deuxième méthode consiste à utiliser une table prédéfinie de 8*8 coefficients de quantification notés $q(u, v)$ et pondérés, là aussi, par le facteur de qualité Q.

La norme JPEG propose deux tables de quantification [Figure 69], une pour les composantes de luminance et l'autre utilisée pour les composantes de chrominance. Ces tables ont été établies à partir d'études psychovisuelles visant à déterminer la sensibilité de l'œil en fonction des fréquences générées.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Figure 69 : *Table de quantification des coefficients de la TCD appliquée à la luminance*

Codage entropique des coefficients de la TCD : une fois quantifiés, les coefficients de la TCD sont traités par un algorithme de compression visant à diminuer encore la quantité de données nécessaires au stockage de la table des coefficients. Comme nous l'avons précisé précédemment, les coefficients de haute fréquence, la plupart du temps nuls, se retrouvent dans le coin inférieur droit de la table des coefficients. C'est pourquoi, afin d'augmenter l'efficacité du codage, les coefficients sont traités en parcourant la table en zigzag [Figure 70].

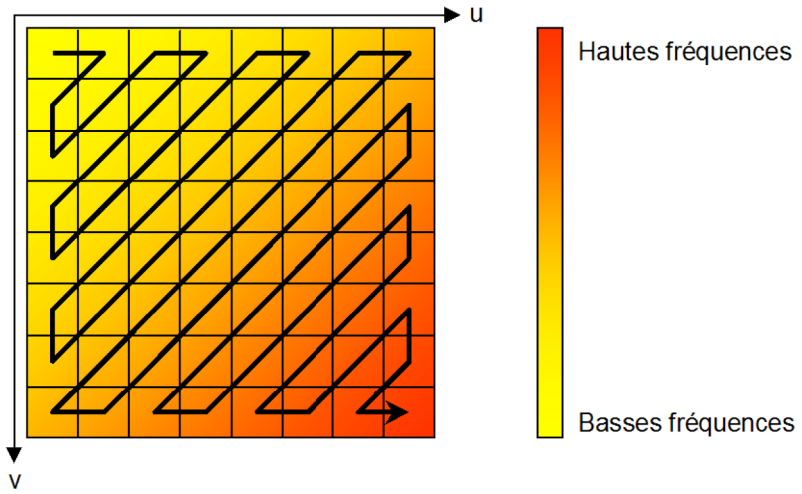


Figure 70 : *Parcours de la table des coefficients fréquentiels en vue de leur compression*

Plusieurs algorithmes de compression des coefficients sont supportés par la norme JPEG (RLE, DPCM, Huffman). Parmi ceux-ci, le codage de Huffman [Jain89] est le plus souvent utilisé car il a l'avantage d'être libre de droits d'utilisation et possède la propriété d'être optimal dans la mesure où il donne la plus petite moyenne de longueur de code de toutes les techniques de codage statistique.

Le format JPEG est intéressant à plus d'un titre. Il offre tout d'abord un taux de compression paramétrable et très performant (de l'ordre de 95 % sans perte notable de qualité). De plus, le temps nécessaire au codage et au décodage d'une image au format JPEG tire profit de l'orientation multimédia des nouveaux processeurs avec, par exemple, le jeu d'instructions MMX d'Intel ou encore les cartes graphiques 2D qui intègrent le plus souvent des fonctions dédiées au traitement de ce format.

Nous avons implanté le codage MJPEG et mesuré ses performances sur un Pentium II cadencé à 266MHz sous Windows95. Nous avons obtenu, pour la compression d'un flux vidéo d'une durée de 1 seconde à 10 Hz, au format 176*144 et en 8 bits, un temps moyen de codage de 40 ms et un temps moyen de décodage de 1 ms. Le débit du flux original [Figure 71] était de 2 Mbits par seconde pour un débit moyen du flux codé [Figure 72] de 128 Kbits par seconde, ce qui reste encore trop élevé pour le type d'application visé. Cependant, cette évaluation a permis de valider la possibilité d'utiliser le codage JPEG en temps réel sans augmenter de façon notable la charge du processeur. Nous allons voir que ce type de codage sert de base aux principales normes pour la transmission de flux vidéo.

Figure 71 : *Image extraite du flux vidéo original*

Figure 72 : *Image extraite du flux MJPEG reconstruit*

Codage H.261

Comme nous venons de le voir, la compression image par image d'un flux vidéo de donne pas de résultats suffisants, en terme de débit généré, pour une application de téléconférence où plusieurs flux, correspondant à plusieurs utilisateurs distants, arrivent simultanément en un même point. Cette observation est toujours valable même lorsque l'on utilise les meilleurs algorithmes de compression d'images fixe tels que le codage JPEG.

Pour répondre à ce problème, le CCITT avait mis en place, dès 1984, le groupe d'étude SG-XV chargé de l'étude d'une norme de compression vidéo pour les applications de visiophonie. A l'origine, le but était de concevoir un codage vidéo pour les transmissions à des débits de m*384 Kbits par seconde ($m \in [1,5]$). Finalement, le standard « Recommandation H.261 : Codec pour Services Audiovisuels à p*64 Kbits par seconde ($p \in [1,30]$) » fut adopté en décembre 1990 afin de couvrir les possibilités de transmission du Réseau Numérique à Intégration de Services (RNIS).

La norme H.261 définit deux formats d'images afin de s'affranchir des problèmes de compatibilité inhérents au différents standards de télévision de par le monde.

Le format CIF : les images ont une résolution de 352*288.

Le format QCIF : les images ont une résolution de 176*144.

Elle introduit, de plus, un mécanisme très important qui permet de prendre en compte la corrélation temporelle des images pour compresser, encore un peu plus, le flux généré. On défini pour cela deux types d'images qui composeront un flux vidéo H.261 [Figure 73] :

Les images I (ou Intra) : ces images sont codées de façon parfaitement autonome à l'aide d'un codeur de type JPEG.

les images P (ou Prédictives) : ces images sont codées à partir des différences relatives à la dernière image **I** générée.

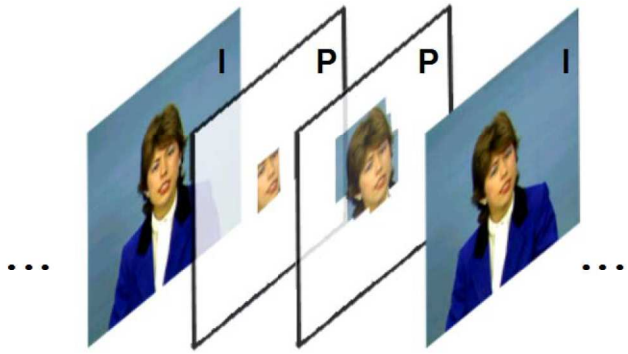


Figure 73 : *Flux vidéo H.261*

Le codage des images prédictives s'effectue en 5 étapes :

Détection du mouvement : le rôle de l'algorithme de détection du mouvement est d'identifier les blocs de l'image en cours de traitement qui sont « suffisamment » différents des blocs correspondants dans l'image précédente. Pour ce faire et dans un souci d'optimisation, on sélectionne, pour chaque bloc de 8*8 pixels, quatre pixels que l'on compare avec leur homologue de l'image précédente. Ces quatre pixels changent d'une image sur l'autre afin de ne pas tester constamment les mêmes zones. On calcule alors une valeur de distance qui détermine si l'on doit marquer le bloc comme ayant subit un changement important. Les blocs ainsi identifiés sont ensuite encodés. On estime que les autres blocs n'apporte aucune information significative par rapport à l'image précédente.

Compensation en mouvement : le rôle de cet algorithme est d'identifier les macro-blocs de 16*16 pixels qui se sont déplacés depuis l'image précédente. La méthode consiste à étudier la présence de pixels semblables dans un voisinage proche et à générer, en cas de succès, le vecteur de déplacement correspondant. La recommandation H.261 ne rend pas obligatoire cette étape de traitement qui comporte deux inconvénients majeurs, à savoir son coût en calcul et le fait qu'elle rend le transfert du flux généré très sensible à la perte de paquets.

Codage par TCD : chaque bloc marqué subi une transformation en cosinus afin d'obtenir sa représentation fréquentielle.

Quantification : les coefficients fréquentiels de la transformée sont simplifiés selon la même méthode employée pour le codage JPEG.

Codage entropique : là encore, la méthode de compression des coefficients reste similaire à celle utilisée par les codeurs JPEG.

Le lecteur pourra se reporter à [Turletti95] pour les détails de la mise en place d'un codeur/décodeur H.261.

Codage H.263

La norme H.263, adoptée en 1996 par l'ITU, constitue une solution efficace aux problèmes exposés dans ce paragraphe. Elle offre en effet des solutions pour le transfert de flux vidéo à un débit inférieur à 64 Kbits. Le principe de fonctionnement d'un codeur/décodeur H.263 est très similaire à son homologue H.261. Cependant, certaines améliorations et autres optimisations permettent d'obtenir, avec cette norme, la même qualité qu'avec la norme H.261 pour un débit moitié moindre :

Structure de données : la structure de données utilisée pour le transfert des images au travers du réseau a été allégée.

Compensation en mouvement : la précision a été améliorée pour la compensation en mouvement afin de pouvoir prendre en compte des vecteurs de déplacement avec une précision d'un quart de pixel.

Formats d'image : H.263 supporte maintenant 3 nouveaux formats d'image (SQCIF 128*96, 4CIF 704*576 et 16CIF 1408*1152).

Un certain nombre d'options négociables entre le décodeur et le codeur ont été introduites :

Compensation en mouvement : un vecteur de déplacement peut maintenant sortir de l'image. Cette option est très utile pour gérer les mouvements de pixels le long des bords de l'image.

Type d'image : un nouveau type d'image est introduit : les images PB. Les images PB sont des images définies par différence entre l'image P précédemment reçue et l'image P en cours de décodage. Cette technique permet de doubler la fréquence des images sans engendrer un trop grand surcoût de débit.

Nous avons choisi d'implanter la norme H.263 dans notre système car elle offre sans conteste le meilleur rapport qualité/débit. Cette opération a été effectuée à partir des sources développées par la société Telenor (http://www.telenor.no). Nous avons évalué les performances de cette solution sur un Pentium II cadencé à 266MHz sous Windows95. Nous avons obtenu, pour la compression d'un flux vidéo d'une durée de 1 seconde à 5 Hz, au format 176*144 et en 8 bits, un temps moyen de codage de 833 ms et un temps moyen de décodage de 210 ms. Le débit du flux original était de 1 Mbits par seconde pour un débit moyen du flux généré de 2,15 Kbits (avec un coefficient de quantification fixé à 25, la qualité étant comparable à celle présentée Figure 72). Le seul inconvénient de cette méthode réside dans sa grande consommation en temps de calcul, impliquant l'utilisation d'une machine bi-processeurs. Néanmoins, nous pensons pouvoir réduire les temps de traitement d'un facteur 5 en utilisant le jeu d'instructions MMX des processeurs Intel.

VIPER offre, en attendant de meilleurs performances, la possibilité d'utiliser la norme MJPEG lorsque le type de réseau utilisé peut supporter le débit généré.

Codage MPEG

Le Moving Picture Experts Group propose plusieurs normes de codage spécialement destinées aux flux audio et vidéo. Les normes MPEG-1 [MPEG1] et MPEG-2 [MPEG2] sont initialement dédiées aux codages de flux de l'ordre de 1.5 et 10 Mo par seconde. Ces normes sont essentiellement destinées à la diffusion de films de qualité télévision ou supérieure. Le codage du flux est similaire à celui utilisé dans la norme H.263. La norme MPEG-4 [Koenen98] devrait, quant à elle, pouvoir être utilisable pour des applications de téléconférence.

4.4.3 Implantation dans le système VIPER

Les mécanismes de codage et de décodage des télécommunications ont été implantés dans les modules « interprète » [Figure 58] [Figure 59] des capteurs et des effecteurs de type flux audio et flux vidéo. Les trames nécessaires sont véhiculées, par les stimuli correspondants, via les espaces de stimuli audio et vidéo, en charge de leur bon acheminement [Torguet98].

4.5 Conclusion

Nous avons présenté, dans ce chapitre, la structure logique qui permet de concevoir une application de réalité virtuelle distribuée avec VIPER. La structure proposée est inspirée des modèles définis pour la simulation comportementale et en particulier du modèle In Vitram (In Vitro Animats) [Rainjonneau92] élaboré dans notre équipe de recherche. Le détail de l'implantation des couches basses de VIPER ainsi que la description de ses aspects répartis pourront être trouvés dans [Torguet98].

Notre contribution à la définition et au développement de cette plate-forme logicielle a principalement porté sur trois points.

Nous avons tout d'abord défini un modèle de comportement basé sur le formalisme des Réseaux de Petri à Objets. L'utilisation d'un tel formalisme permet de concevoir, de manière souple et intuitive, et de valider le comportement des entités. A ce sujet, nous ne disposons pas encore d'un éditeur graphique de comportements qui pourrait simplifier, encore un peu plus, leur construction et leur validation. Il serait sans aucun doute très utile de disposer d'un tel outil dans un avenir proche. Nous avons ensuite proposé le concept de niveaux de détails de comportement permettant d'adapter la complexité de ce dernier en fonction de l'importance de l'entité dans l'environnement, cette importance étant pour l'instant déterminée selon un critère purement graphique, à savoir sa taille à l'écran.

Nous avons ensuite défini et optimisé les mécanismes utilisés par VIPER pour la détection des collisions en introduisant le concept de cache de collisions qui permet d'éviter de nombreux calculs inutiles lorsque deux entités immobiles s'interpénètrent. De plus, cette méthode permet de tirer profit de l'aspect distribué de la simulation en mettant les résultats des calculs de collision à la disposition de tous les utilisateurs.

Enfin, nous avons étudié et implanté les mécanismes nécessaires à la communication audio et vidéo entre les différents utilisateurs impliqués dans une session de travail coopératif. Les différentes solutions envisagées, choisies parmi les standards en vigueur, ont été évaluées afin de déterminer leur pertinence dans le contexte temps réel du type d'application visé. Pour donner un ordre de grandeur, la gestion des flux audio (GSM) et vidéo (MJPEG) consomme, pour une session VIPER impliquant quatre utilisateurs, environ 12% du temps machine sur un Pentium II cadencé à 266 MHz, le reste du temps étant disponible pour la simulation temps réel.

Nous allons présenter, dans le chapitre suivant, l'architecture et l'implantation d'un système de prototypage virtuel coopératif basé sur VIPER.

5 Le système PROVIS

Nous avons présenté, dans le chapitre précédent, le système VIPER dédié au développement d'applications de réalité virtuelle distribuée. Nous avons ainsi décrit son architecture, son modèle comportemental ainsi que les fonctionnalités dont il dispose pour permettre la gestion de sessions de travail coopératif.

C'est sur cette plate-forme qu'a été développé le système pour le prototypage virtuel coopératif, PROVIS, que nous allons décrire dans ce chapitre. Nous présenterons, au travers des entités qui le composent, son architecture générale ainsi que ses fonctionnalités en termes d'interaction naturelle, de dialogue multimodal et de travail coopératif. Nous terminerons, enfin, par la présentation du modèle de conception que nous proposons pour l'assemblage de prototypes et la création de mécanismes complexes.

PROVIS [Balet97] est défini par quatre principaux types d'entité [Figure 74], à savoir, les avatars, le Constructeur, l'Expert et les prototypes.

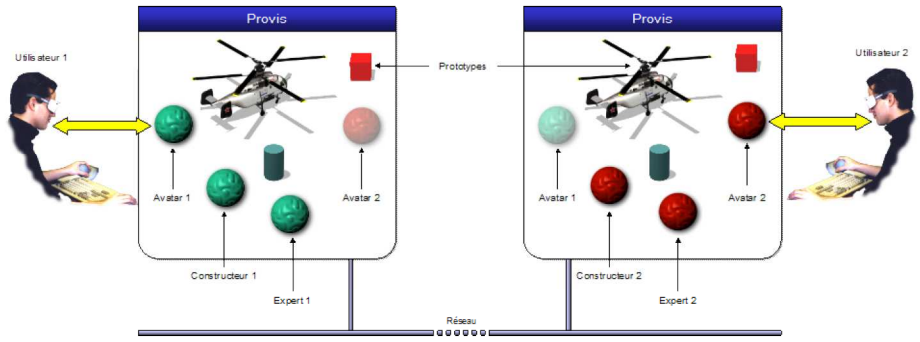


Figure 74 : *Architecture générale d'une session PROVIS*

5.1 Les avatars

Ces entités représentent les utilisateurs qui partagent la session de travail. L'une d'entre elles est ainsi le clone [Duthen93] [Mouli94] de l'utilisateur local. La connexion d'un utilisateur au système PROVIS engendre automatiquement la création d'un tel clone défini par un nom d'utilisateur, des droits d'accès, une URL (Uniform Ressource Location), une fenêtre de visualisation ainsi qu'une fenêtre de gestion des communications toutes deux locales à la station utilisée. Cet avatar [Figure 75] comporte plusieurs modules comportementaux ainsi qu'un ensemble de capteurs et d'effecteurs lui permettant d'émettre des ordres vers d'autres entités, de visualiser l'environnement 3D et d'entrer en communication avec les autres utilisateurs. Un avatar local possède, de plus, des capteurs directement connectés aux périphériques du système.

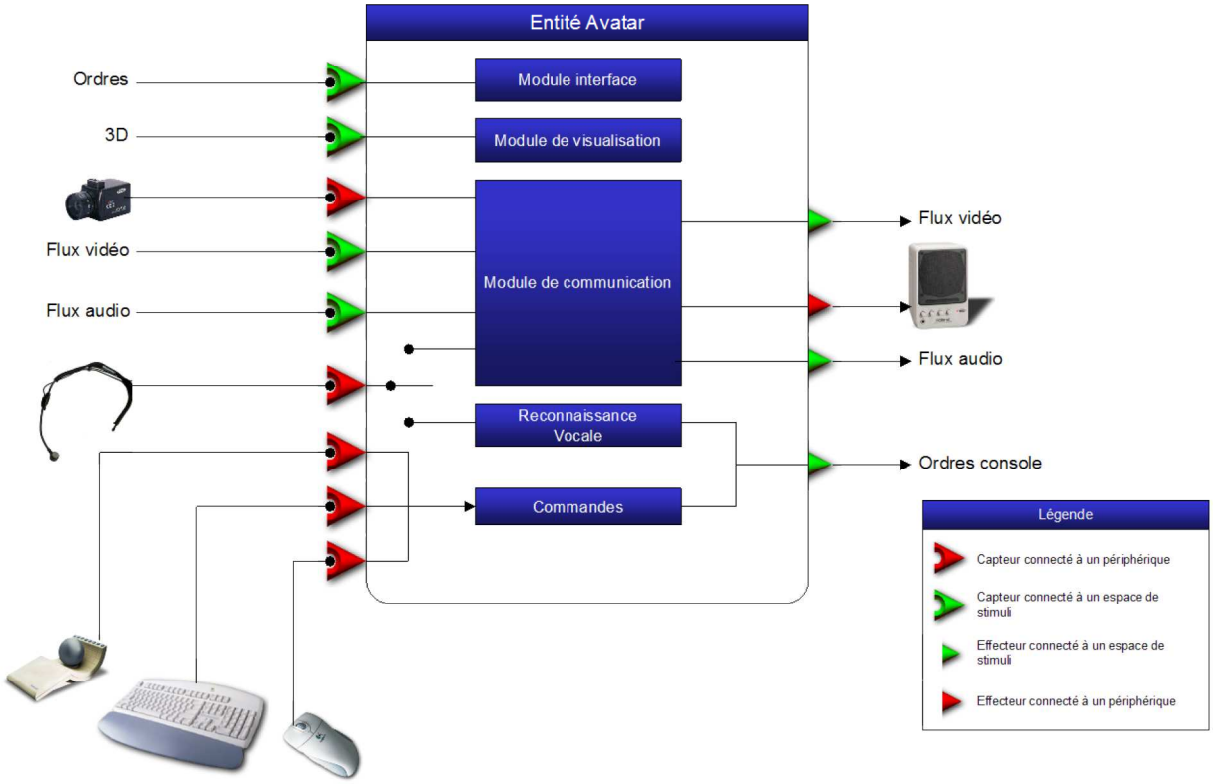


Figure 75 : *Architecture d'une entité avatar*

Nous avons choisi de ne pas donner de forme tridimensionnelle aux clones des utilisateurs distants afin d'éviter l'ajout de nouveaux objets dans l'environnement visualisé. En effet, la représentation tridimensionnelle d'utilisateurs distants gène très souvent le processus de modélisation en surchargeant la visualisation, masquant parfois la zone de travail. Un avatar distant est donc représenté par une icône (ou un flux vidéo), placé dans la fenêtre de communication, ainsi que par un flux audio dans le cas, bien sûr, où il en émettrait un.

5.1.1 Le module de visualisation

Ce module est en charge de gérer la fenêtre de visualisation 3D [Figure 76] grâce à laquelle l'utilisateur perçoit une vue tridimensionnelle de l'environnement. Ce module comportemental reçoit des stimuli d'un capteur de type caméra (noté « 3D » [Figure 75]) qui lui permet de « voir » les autres entités ainsi que le décor qui composent l'environnement virtuel.

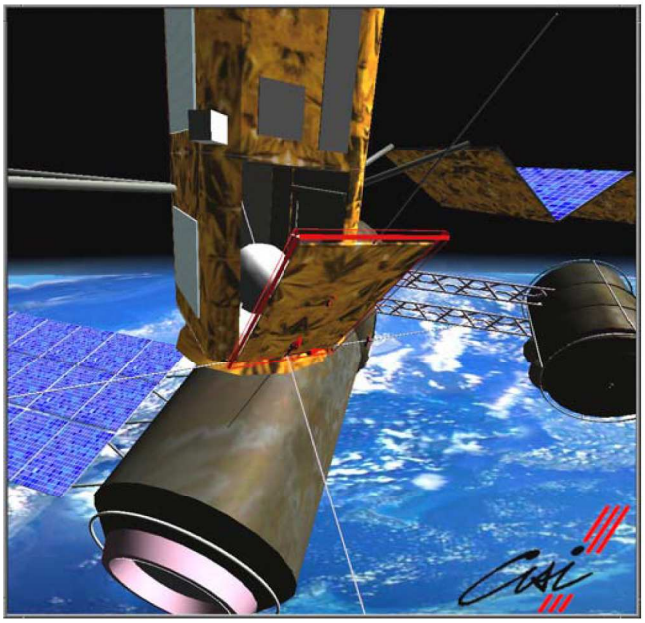


Figure 76 : *Fenêtre de visualisation 3D*

Il existe deux méthodes pour modifier la position et l'orientation du capteur caméra dans l'espace afin de changer le point de vue.

Manipulation à l'aide du contrôleur 3D : l'utilisateur a la possibilité de déplacer la caméra à l'aide du contrôleur 3D. Ce mode de contrôle est utilisé lors des phases de revue du prototype virtuel.

Manipulation à l'aide de la souris : c'est le mode de fonctionnement par défaut. Il repose sur deux métaphores d'interaction nommées « focus on interest » et « inspect mode ».

La première consiste à déterminer l'orientation du point de vue en fonction de l'activité de l'utilisateur. Par exemple, lors du chargement d'un objet, celui-ci se place, de manière autonome, à un endroit de la scène aussi proche que possible de l'endroit que regarde l'utilisateur. Il arrive néanmoins qu'il n'y ait là pas assez de place, obligeant alors l'objet à se placer hors du champ de vision. Dans ce cas, le système recentre le champ de vision sur l'objet chargé. Nous avons ainsi défini un ensemble de règles qui déterminent le centre d'intérêt en fonction des opérations effectuées.

Le principe du mode « inspect » consiste, quant à lui, à contraindre les mouvements de la caméra sur la surface d'une sphère englobant le centre d'intérêt de l'utilisateur, tout en conservant ce dernier au centre du champ de la caméra. Le rayon de cette sphère est modifiable, de manière interactive, en déplaçant la souris, d'avant en arrière, tout en maintenant son bouton central enfoncé. C'est ainsi que l'on génère des effets de zoom qui seront, dans un futur proche, associés à la roulette qui équipe les nouvelles souris.

5.1.2 Le module de gestion des communications

Ce module est en charge de la gestion des communications avec les utilisateurs distants et l'ordinateur. Pour cela, il dispose des informations (flux audio et vidéo, locaux et distants) en provenance des capteurs de l'entité [Figure 75] ainsi que d'une fenêtre d'interface [Figure 77] découpée en cadres. Chacun des utilisateurs distants se voit attribuer un cadre ainsi qu'une couleur.

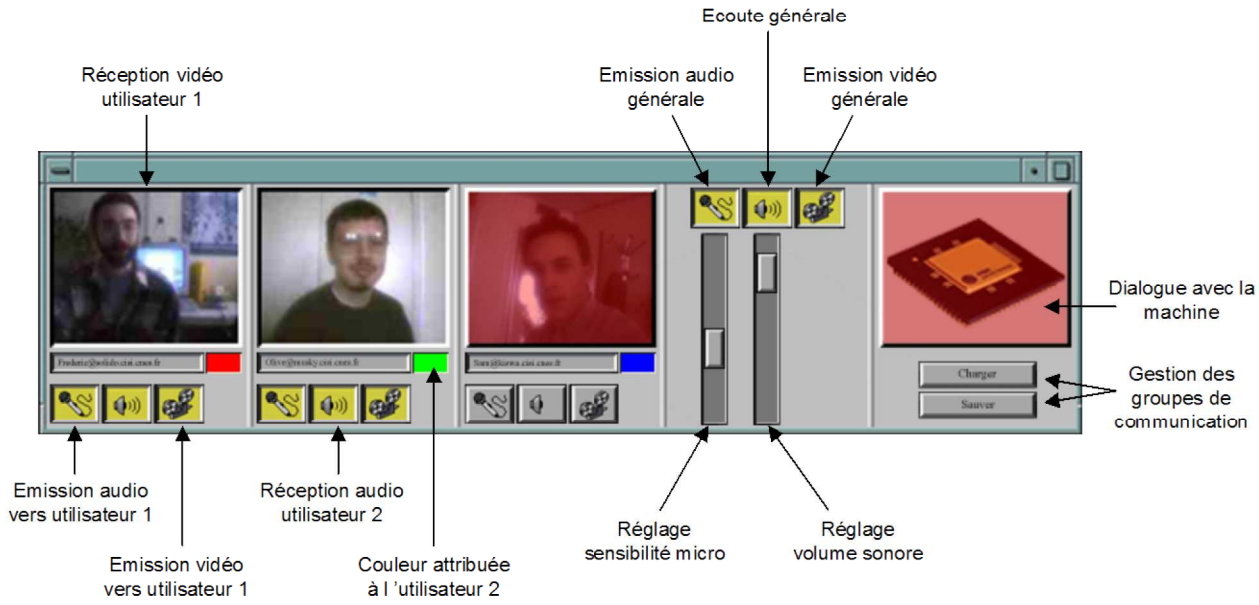


Figure 77 : *Gestion des communications avec les utilisateurs distants*

Ce cadre contient différents éléments d'interface (widgets) permettant la gestion des flux audio et vidéo propres à l'utilisateur représenté. L'utilisateur local peut ainsi régler, dans chaque cadre, les paramètres pour :

La réception audio : il est possible, à l'aide de ce bouton, de ne plus écouter un utilisateur distant.

L'émission audio : ce bouton permet de faire en sorte que l'utilisateur distant n'entende plus les propos de l'utilisateur local.

La réception vidéo : il est possible de geler la réception d'un flux vidéo en cliquant sur l'élément d'interface où il est affiché. Dans ce cas, le module affiche la dernière image reçue à laquelle il rajoute un cache translucide (utilisateur 3 [Figure 77]).

L'émission vidéo : l'utilisateur local peut stopper l'envoi de son flux vidéo vers un utilisateur distant.

Il est possible de mémoriser la configuration des paramètres décrits ci-dessus en créant un groupe de communication qui pourra être rappelé ultérieurement. C'est ainsi que, par exemple, l'utilisateur peut passer très rapidement d'une situation d'aparté à une situation de dialogue impliquant certains des utilisateurs.

La fenêtre de gestion des communications dispose, de plus, d'un ensemble d'éléments d'interface permettant le contrôle général des périphériques de communication (volume sonore du haut-parleur, sensibilité du micro, bouton « mute » micro, bouton « mute » haut-parleur, bouton « mute » caméra).

Une zone de la fenêtre est réservée au dialogue avec la machine. Notre système pouvant en effet fonctionner avec un système de reconnaissance vocale, il est indispensable de proposer une solution pour déterminer à qui s'adresse l'utilisateur lorsqu'il parle. Nous proposons de considérer le dialogue avec la machine comme une situation d'aparté. L'utilisateur qui souhaite s'adresser à la machine doit choisir explicitement ce mode en cliquant sur l'icône correspondante [Figure 77]. Cette action a pour effet de sauvegarder la configuration de communication courante et de désactiver l'envoi du flux audio vers l'ensemble des utilisateurs. Par la suite, l'utilisateur peut reprendre la communication avec ses partenaires en cliquant à nouveau sur l'icône de l'ordinateur, la configuration de communication sauvegardée étant alors restaurée. Le module de reconnaissance vocale fournit, en pratique, la traduction des propos de l'utilisateur sous forme d'une chaîne de caractères.

C'est lors des phases d'évaluation du système PROVIS que nous avons découvert que la communication entre utilisateurs distants pouvait être améliorée en proposant un dispositif permettant d'attacher le point de vue local à un point de vue distant. Ainsi, deux utilisateurs peuvent partager la même référence visuelle, éliminant ainsi un grand nombre d'ambiguïtés et de problèmes de compréhension lors du dialogue. Pour compléter ce mécanisme, l'utilisateur local qui décide de prendre le point de vue d'un utilisateur distant (en cliquant sur son icône de couleur [Figure 77]), voit s'afficher le pointeur souris de ce dernier, la couleur de ce pointeur correspondant à la couleur de l'utilisateur choisi. C'est ainsi que des expressions du type « Qu'est-ce que tu penses si je mets ça ici ? » prennent tout leur sens.

5.1.3 Le module de reconnaissance vocale

Nous avons utilisé le système DATAVOX de la société VECSYS pour assurer la reconnaissance de la parole. Ce système fonctionne en mode mono-locuteur et en environnement non bruité. Il est capable de reconnaître un flux de parole continu et de le transformer en une chaîne de caractères. Il nécessite pour cela la définition d'une grammaire définissant le langage d'interaction entre un usager et le système PROVIS. Plus cette grammaire est rigide et meilleure est la qualité de reconnaissance. C'est pourquoi nous avons défini un langage de taille limité (moins de 500 mots) que nous décrirons ultérieurement (cf. paragraphe 5.2.2). Le lecteur pourra se reporter à [Brison97] pour obtenir les détails de l'implantation de ce langage sur le système DATAVOX.

Ce module fournit donc en sortie des chaînes de caractères dont chaque mot est daté.

5.1.4 Le module interface 2D

Le module interface reçoit des stimuli, de type ordre, au travers des capteurs de l'avatar. Il est en charge de l'interface 2D permettant à l'utilisateur de gérer la session de travail. Cette interface n'est en fait qu'un simple cadre 2D [Figure 78] où les entités viennent rajouter ou enlever, en envoyant des stimuli de type ordre, les éléments d'interface qui leur sont propres [Figure 79]. Conceptuellement, un élément d'interface représente un capteur (boutons, champs de saisie) ou un effecteur (champs de valeur) d'une entité.

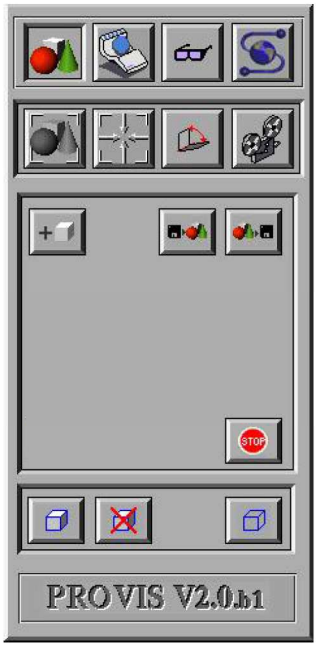


Figure 78 : *Interface 2D contenant les widgets de l'entité Constructeur*

Figure 79 : *Une entité sélectionnée vient y rajouter ses propres widgets*

Son fonctionnement est pour l'instant basé sur la bibliothèque X-FORMS. Nous avons choisi cette bibliothèque de construction d'interfaces pour sa portabilité et surtout pour la disponibilité de ses sources. En effet, nous avons dû modifier ces dernières pour permettre le travail en vision stéréoscopique sur des stations bas de gamme qui entrelacent les images pour produire un rendu en relief. Ce procédé consiste à découper verticalement l'écran en deux parties, chacune d'elles étant destinée à contenir l'image pour l'un des deux yeux de l'utilisateur. Ces images sont ensuite affichées, alternativement sur l'écran, en synchronisation avec des lunettes à obturateur qui masquent l'œil auquel n'est pas destinée l'image courante. Il est alors impossible d'utiliser un système d'interface classique, tel que X-Windows ou Windows, dont les widgets se retrouvent, ou présentes sur une seule des deux images, ou découpées [Figure 80], engendrant ainsi un inconfort d'utilisation pour le moins certain.

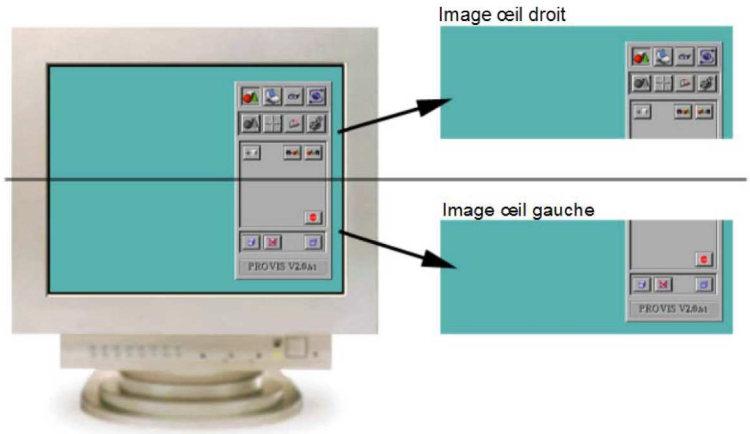


Figure 80 : *Découpage d'une interface classique en mode stéréoscopique entrelacé*

Pour éviter ce type de phénomène et plutôt que de créer deux interfaces qu'il nous aurait fallu synchroniser, nous avons préféré modifier les sources de cette bibliothèque afin de doter chaque widget d'une double représentation [Figure 81]. Chaque élément d'interface ne possède donc plus une seule mais deux représentations graphiques correctement situées, fournissant ainsi une vue parfaitement recalée de l'interface graphique. De la même manière, les événements (clics souris, enfoncement de boutons) sont traités de telle sorte qu'ils produisent le même effet sur les deux représentations graphiques. Cette solution est particulièrement élégante puisque le développeur de l'interface ne se soucie pas de savoir dans quel mode elle fonctionnera (monoscopique, stéréoscopique). C'est le noyau de la bibliothèque qui assure, de manière transparente, son bon fonctionnement en mode stéréoscopique.

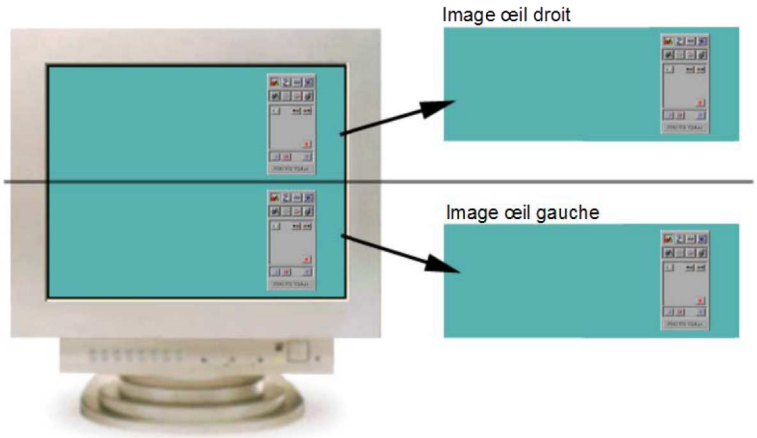


Figure 81 : *Dualité des éléments d'interface*

La justification de l'existence du module interface 2D peut être contestée d'un point de vue conceptuel. En effet, nous pensons qu'il serait plus judicieux de considérer l'ensemble des widgets d'une entité comme une autre représentation graphique de sa forme.

Dans ce cas, le système VIPER devrait alors proposer la possibilité d'obtenir simultanément plusieurs vues d'une même entité. Un utilisateur pourrait ainsi disposer, par exemple, de la vue tridimensionnelle d'une entité dans la fenêtre de visualisation 3D et de la vue 2D de cette même entité sous la forme d'un ensemble de widgets. C'est d'ailleurs dans cette direction que nous faisons actuellement évoluer VIPER.

5.1.5 Modes d'interaction

Figure82 : *Interaction multimodale*

Nous avons pris le parti de ne proposer qu'une seule fenêtre de visualisation alors que la quasi totalité des outils de modélisation en proposent quatre (haut, face, coté, perspective). Compte tenu l'importance de l'aspect ergonomique de notre système qui, pour rappel, s'adresse à des utilisateurs non experts, nous avons mené une campagne d'études afin de valider la pertinence de nos choix. Pour cela, une trentaine de personnes ont été consultées et se sont prêtées à nos tests. Ces personnes ont été classées en trois groupes :

Les experts : il s'agit en fait des personnes, au nombre de 8, ayant une pratique courante des outils de modélisation 3D et des techniques d'interaction tridimensionnelle.

Les initiés : il s'agit des personnes, au nombre de 9, ayant une pratique courante de l'informatique graphique (traitement d'images, modèles numériques de terrain, etc.) ou des jeux vidéo. Ces personnes possèdent donc les notions nécessaires pour appréhender le travail en trois dimensions sur un écran.

Les novices : le groupe des novices est constitué de 11 personnes ayant l'habitude de travailler avec les outils classiques de l'informatique (souris, fenêtres) mais n'ayant aucune expérience des environnements tridimensionnels.

Deux types de contrôleurs 3D, de fonctionnement similaire, ont été utilisés afin de déterminer lequel d'entre eux offrait la meilleure ergonomie :

La Spaceball : ce contrôleur dispose d'une sphère de contrôle fixe que l'utilisateur ne peut bouger lors de ces manipulations.

Figure 83 : *Le contrôleur Spaceball de Spacetek*

La Spaceball : ce contrôleur dispose d'un palet de contrôle légèrement mobile permettant à l'utilisateur d'appréhender physiquement ses manipulations.

Figure 84 : *Le contrôleur Spacemouse (ou Magellan) de Logitech*

L'objectif premier de nos tests était de déterminer si l'utilisation d'une seule fenêtre, associée au mode « inspect », permettait de manipuler et de placer les objets plus rapidement qu'avec une solution traditionnelle proposant quatre vues. Pour ce faire, nous avons demandé à chaque candidat de placer et d'orienter un objet tridimensionnel de manière à le faire correspondre avec une silhouette, placée à un endroit fixe de l'espace. Cette expérience a été répétée 10 fois pour chaque configuration de contrôleur, avec et sans vision stéréoscopique et pour chacune des personnes. La configuration de contrôleur est définie par le type de contrôleur 3D utilisé ainsi que par son mode de fonctionnement ; il existe en effet deux modes différents. Le mode dominant ne prend en compte que le mouvement de l'utilisateur qui a la plus grande amplitude (translation sur x, y ou z ou rotation autour de l'un de ces trois axes). Le mode non dominant, quant à lui, prend en compte la totalité des mouvements.

Nous présentons, [Tableau 3], les résultats de cette campagne d'évaluation.

Rendu	Contrôleur	Contrôle	Experts		Initiés		Novices	
			Temps[1]	Note[2]	Temps	Note	Temps	Note
Monoscopique 1 vue	Spaceball	Dominant	21,1	6	36,2	4	56,3	3
		Non dominant	18,8	7	36,5	4	62,5	2
	Spacemouse	Dominant	30,3	5	35,3	5	54,3	5
		Non dominant	26,0	6	35,4	5	65,8	4
Stéréoscopique 1 vue	Spaceball	Dominant	8,7	7	21,5	4	39,3	4
		Non dominant	8,5	8	22,1	4	45,1	3
	Spacemouse	Dominant	16,3	4	21,6	7	32,1	7
		Non dominant	11,8	5	21,4	7	50,5	5
Monoscopique 4 vues	Spaceball	Dominant	29,1	5	33,1	5	48,1	4
		Non dominant	30,5	4	35,5	4	57,6	3
	Spacemouse	Dominant	32,2	4	34,3	6	45,6	6
		Non dominant	35,2	3	35,0	6	59,1	5

Tableau 3 : *Résultats de la campagne d'évaluation de l'ergonomie de notre système*

[1] Temps de réalisation de l'expérience exprimé en secondes

[2] Degré de satisfaction de l'utilisateur noté sur 10 points

Plusieurs enseignements peuvent être tirés des résultats obtenus. Nous avons tout d'abord pu montrer que l'utilisation d'une seule fenêtre de rendu, associée au mode « inspect », offrait le même niveau d'ergonomie qu'un système de visualisation traditionnel pour des utilisateurs un minimum entraînés. Seules les personnes novices semblent mieux se débrouiller avec quatre vues. Nous pensons que leur principal problème réside dans le fait que le mode « inspect » est de type multimodal, c'est à dire qu'il attribut la manipulation du point de vue à la souris et la manipulation des objets au contrôleur 3D. L'utilisation synergique de ces deux périphériques permet d'assurer les meilleurs résultats alors qu'un utilisateur novice est peu habituer à manipuler simultanément deux périphériques.

Nous avons ensuite pu confirmer que la visualisation stéréoscopique de l'environnement permettait d'accélérer grandement les tâches de manipulation [Kim87]. Cette accélération est d'autant plus importante que le degré d'expertise de l'utilisateur est élevé. On constate généralement un gain de temps allant de 30 à 70%. Il est à noter que nous n'avons pas utilisé, dans cette campagne d'études, de système permettant de localiser la position de la tête de l'observateur et de générer ainsi les effets de parallaxe. Malgré le faible nombre d'études portant sur ce domaine, il est montré [Richard98] [Ware93] [Ikehara93] que les effets de parallaxe permettent de faciliter la perception de la profondeur et d'accélérer ainsi les tâches de manipulation en 3D. A ce sujet, nous poursuivons actuellement notre campagne de mesures en utilisant le système CrystalEyesVR afin de quantifier le gain de temps que peut apporter un tel système.

Enfin, nous pouvons constater que les utilisateurs les moins expérimentés éprouvent des difficultés à utiliser la Spaceball. En effet, le caractère figé de la sphère de contrôle de ce périphérique engendre des phénomènes de crispation et donc de fatigue musculaire, les utilisateurs ayant tendance à forcer sur le mécanisme. En revanche, ce périphérique permet des manipulations plus précises pour les utilisateurs les plus avertis. La Spacemouse, quant à elle, semble mieux convenir au grand public mais n'offre pas la même précision de manipulation que sa concurrente. De plus, si le mode dominant semble aider les utilisateurs novices, il semble que le mode non dominant soit le mieux adapté pour une utilisation « professionnelle ».

Il est à noter que l'utilisation d'un contrôleur 3D intégrant le retour d'effort pourrait améliorer grandement la qualité de l'interaction. Ce type de dispositif permettrait, par exemple, de ressentir les forces générées par les contacts entre les divers objets manipulés. Malheureusement, nous ne disposons pas d'un tel périphérique dans notre laboratoire. Son intégration dans notre système ne devrait cependant pas poser de problème conceptuel dans la mesure où il est possible de rajouter un effecteur haptique à l'entité avatar. Par contre, le haut débit des informations qu'il faut fournir pour restituer un retour d'effort correct (>1000Hz [Cadoz90] [Burdea93]) nécessiterait une étude approfondie pour pouvoir intégrer ce type de dispositif dans un système distribué tel que le notre.

5.2 Le Constructeur

Cette entité est en charge du dialogue naturel multimodal avec l'utilisateur, via son avatar, afin de lui fournir un ensemble de commandes de haut niveau lui permettant de gérer, manipuler et assembler les prototypes. C'est donc cette entité qui définit les fonctionnalités du système PROVIS.

Une entité de type Constructeur accompagne chaque avatar localement et n'est donc pas distribuée dans l'environnement virtuel ; elle n'a qu'une existence locale à la station de travail [Figure 74].

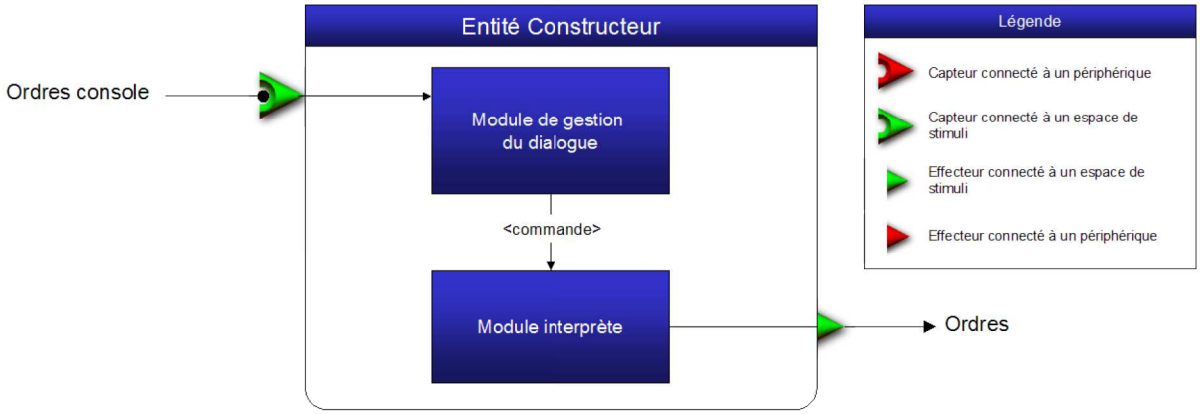


Figure 85 : *Architecture générale de l'entité Constructeur*

Cette entité possède deux modules comportementaux chargés, l'un du dialogue avec l'avatar de l'utilisateur, l'autre de l'interprétation et de l'exécution des commandes. Elle utilise, en outre, un capteur d'ordres console qui lui permet de recevoir des stimuli de type « texte », « souris » et « contrôleur 3D ». Nous avons choisi d'unifier ces trois types de stimulus sous un type plus général « console » afin de ne pas alourdir inutilement le nombre de capteurs à utiliser pour recevoir les informations en provenance de l'utilisateur. L'entité Constructeur utilise, de plus, un effecteur d'ordres pour envoyer des commandes aux entités comme, par exemple, pour installer des widgets dans l'interface graphique de l'avatar ou pour effectuer des modifications sur une entité prototype.

5.2.1 Dialogue multimodal

Le concept de multimodalité [Lewis91] [Thorisson92] [Brison97] est né de la multiplication et de la diversification des périphériques d'entrée et de sortie de données. L'idée consiste à doter une application, une entité dans notre système, de plusieurs modalités d'entrée qui pourront être combinées entre elles pour énoncer une commande.

On désigne par modalité un moyen de communication d'un point de vue humain. Les modalités d'entrée peuvent être la parole, l'écrit (langage naturel), le geste (désignation, déformation, commande, langage gestuel), la direction du regard, etc. Les modalités de sortie peuvent être de type sonore (parole codée, parole à partir de texte, musique), visuel (graphique, texte), tactile ou haptique (retour d'effort), etc.

On appelle énoncé l'ensemble des événements utilisateur qui ont été nécessaires pour produire une commande. Un énoncé peut être composé soit d'un geste de désignation, soit d'une parole ou de toute autre combinaison des modalités disponibles. Un énoncé monomodal est par conséquent composé d'événements utilisateur provenant d'une seule modalité d'entrée. Par analogie, un énoncé multimodal est composé de plusieurs événements utilisateur dont les modalités d'entrée sont différentes.

Par exemple, un utilisateur peut donner oralement la commande « Pose ça là » tout en indiquant, à l'aide de la souris, ce que représentent les déictiques (références intermodales) « ça » et « là ». L'identification des déictiques ainsi que la construction d'une seule commande à partir de l'énoncé complet est un mécanisme appelé fusion multimodale.

Tous les systèmes interactifs disposant d'une interface multimodale ne sont pas identiques. Ils peuvent être classés [Tableau 4] suivant le type de multimodalité supportée qui est essentiellement défini par trois paramètres suivants :

La production des énoncés : elle peut être séquentielle ou parallèle.

Le nombre de médias par énoncé : un ou plusieurs médias peuvent être utilisés pour construire l'énoncé.

L'usage des médias : ces médias peuvent être utilisés de manière exclusive (un à la fois) ou simultanée (synergique).

Les interfaces multimodales purement exclusives ne posent pas de problèmes quant à la fusion d'informations ou à la résolution de références intermodales : chaque média est utilisé de manière indépendante.

Par contre, les interfaces multimodales alternées ou synergiques posent des problèmes de relation entre ces modalités (cf. paragraphe 5.2.4.3) ainsi que des problèmes de fusion (cf. paragraphe 5.2.4.4) des informations contenues dans ces différentes modalités.

Type de multimodalité	Production des énoncés	Nombre de médias par énoncé	Usage des médias
Exclusive	séquentielle	un	exclusif
Alternée	séquentielle	plusieurs	exclusif
Synergique	séquentielle	plusieurs	simultané
Parallèle exclusive	parallèle	un	exclusif
Parallèle simultanée	parallèle	un	simultané
Parallèle alternée	parallèle	plusieurs	exclusif
Parallèle synergique	parallèle	plusieurs	simultané

Tableau 4 : *Les différents types de multimodalité [Brison97]*

Dans notre système, la production des énoncés se fait de façon séquentielle. Par contre, l'utilisateur peut se servir simultanément de plusieurs médias afin d'énoncer sa commande. Il pourra par exemple déclarer oralement « supprime ça » tout en désignant « ça » à l'aide de la souris. Le dialogue que nous souhaitons traiter est donc de type synergique.

5.2.2 Création d'un langage

La gestion du dialogue naturel, entre l'utilisateur et l'entité Constructeur, nécessite tout d'abord la création d'un langage. Nous limiterons volontairement celui-ci à un faible vocabulaire (inférieur à 500 mots) et aux actions de base (créer, supprimer, déplacer, poser, etc.).

La structure d'une commande de ce langage sera du type « impératif + complément(s) » comme, par exemple : « pose l'antenne sur le support »

> impératif : pose
> complément 1 : l'antenne
> complément 2 : sur le support

Nous avons, dans un premier temps, dressé une liste précise de tous les mots qui peuvent être utilisés dans le cadre de la construction de prototypes virtuels. Cette liste est composée de verbes (à l'impératif), de noms, d'articles (définis et indéfinis), de prépositions, de conjonctions de coordination et de pronoms. Il est à noter que les noms des différents composants de prototypes, propres à chaque métier, ne font pas partie du langage défini. Ils sont rajoutés dynamiquement après avoir été extraits des fichiers VRML décrivant les géométries, assurant ainsi notre système d'une grande indépendance vis à vis du domaine d'utilisation.

Le langage propose également des commandes dites 'floues' [Djedi91] [Dubois97], l'objectif étant de permettre à l'utilisateur de spécifier une commande imprécise comme, par exemple, « déplace l'antenne vers la gauche » ou « vers le bas », etc. Le déplacement se fait alors par rapport au point de vue de l'utilisateur d'une distance qui est calculée en fonction de la taille de l'objet. L'utilisateur peut ensuite corriger le résultat obtenu grâce à des commandes du type « encore », « un peu plus » ou « un peu moins ». La commande « encore » a pour effet de répéter la dernière action, « un peu plus » et « un peu moins » exécute à nouveau la commande précédente en modifiant ses paramètres.

L'utilisateur a également la possibilité d'utiliser des pronoms afin d'effectuer des opérations sur les objets sur lesquels il vient d'agir. Il peut, par exemple, énoncer « déplace le panneau solaire vers la droite, texture le », l'analyseur syntaxique remplaçant alors le pronom « le » par « le panneau solaire ».

L'analyse syntaxique d'un énoncé en langage naturel est réalisée par l'entité Constructeur à l'aide d'un analyseur généré avec le compilateur Yacc (Yet Another Compiler Compiler). Ce programme permet de compiler une grammaire du type LALR(1) afin de produire le texte source d'un analyseur syntaxique du langage engendré par cette grammaire. La définition de l'analyseur lexical a été, quant à elle, réalisée à l'aide de l'outil Lex.

5.2.3 Normalisation des stimuli

Comme nous l'avons indiqué précédemment, l'entité Constructeur supporte le dialogue multimodal avec l'utilisateur. Pour cela, il est nécessaire de disposer d'un formalisme capable de représenter, de façon homogène, les différents types de stimuli reçus. Ainsi, le capteur d'ordres console de l'entité Constructeur est en charge de la transformation des types de stimulus « texte », « souris » ou « controleur3D », en un événement normalisé contenant toutes les informations nécessaires à la gestion du dialogue naturel.

Le formalisme retenu pour la normalisation des stimuli est la structure de traits, un trait étant un couple (identificateur, valeur) dont la valeur peut être de type atomique (nombre, caractères) ou structure de traits.

Un événement normalisé est donc de la forme :

Date :	*date de création du stimulus*
Modalité :	*modalité de création*
Verbe :	*verbe à l 'impératif*
Action :	*Action correspondant au verbe (certains verbes peuvent avoir plusieurs sens en fonction des compléments et des propositions utilisées*
Compléments :	*Liste de compléments*
Connecteurs :	*Liste de prépositions ou conjonctions*

Figure 86 : *Structure de traits définissant un message normalisé*

Les compléments sont eux aussi définis par une structure de traits [Figure 87].

Date :	*date de génération*
Catégorie :	*entité, texture, liaison mécanique, couleur, etc.*
Fonction :	*direction, lieu, moyen, manière, complément d'objet direct ou indirect*
Article :	*défini, indéfini, démonstratif, déictique*
Nom :	*nom du syntagme*
Adjectifs :	*adjectifs du syntagme*
Nombre :	*singulier, pluriel*
Genre :	*masculin, féminin*
Indice :	*indice d 'apparition dans l 'énoncé*
Longueur :	*nombre de mots du syntagme*
Valeur :	*trait générique*

Figure 87 : *Structure de traits définissant une valeur du trait complément*

Cette structure comporte un trait générique « *Valeur* » qui contient la valeur du complément. Si le complément est un réel, un vecteur ou un quaternion alors sa valeur sera égale à, respectivement, ce réel, ce vecteur ou ce quaternion. Si le complément est une entité, sa valeur sera alors soit une adresse mémoire pointant sur l'entité en question si elle se trouve déjà en mémoire, soit une URL dans le cas contraire.

Prenons, par exemple, le stimulus de type texte suivant : « Pose l'élément bleu sur ce support ». On obtient, après normalisation, la structure de traits décrite [Figure 88]. De la même manière, la sélection d'un élément à la souris génère l'envoi d'un stimulus à l'entité Constructeur dont la forme normalisée est présentée [Figure 89].

Le capteur d'ordres console ne peut pas toujours compléter l'ensemble des traits de la structure générée. Par exemple, dans le cas d'un stimulus souris, on ne dispose que de l'adresse de l'objet sélectionné. Dans ce cas, la valeur des traits inconnus est fixée à VOID et sera complétée ultérieurement par le module de dialogue.

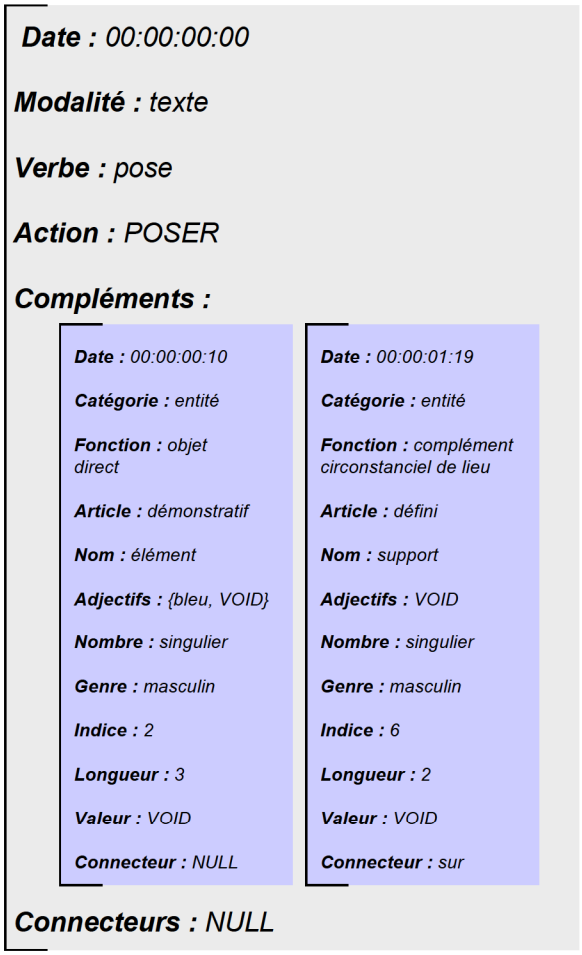


Figure 88 : *Forme normalisée du stimulus textuel « Pose l'élément bleu sur ce support »*

Figure 89 : *Forme normalisée d'un stimulus de type sélection souris*

5.2.4 Module de gestion du dialogue

Ce module comportemental est tout d'abord en charge de vérifier que l'utilisateur possède bien les droits d'accès aux entités qu'il référence dans son énoncé (c.f. paragraphe 5.4.1). Si tel est le cas, ce module complète alors les structures de traits fournies par le capteur. Il peut, pour cela, accéder à l'ensemble des prototypes de l'environnement ainsi qu'à l'historique du dialogue. Une fois les structures complétées, ce module se charge ensuite de les fusionner pour ne plus former qu'une seule commande qui sera transmise au module interprète.

5.2.4.1 Historique du dialogue

L'utilisateur a la possibilité d'utiliser des pronoms et de faire ainsi référence à des entités qu'il a mentionnées auparavant. Il peut, par exemple, énoncer la commande « charge un réservoir » puis, un moment plus tard, indiquer « pose le sur le support ». Il est donc nécessaire de conserver un historique du dialogue afin de pouvoir retrouver l'entité référencée. C'est pourquoi chaque structure de traits, une fois complétée, est sauvegardée dans une liste stockée en mémoire. C'est aussi grâce à ce mécanisme que l'utilisateur peut annuler ou répéter l'énoncé précèdent sans avoir à le reformuler.

5.2.4.2 Complétion des structures normalisées

C'est lors de cette étape qu'est déterminée la valeur des traits laissés vides par le capteur lors de la phase de normalisation. Reprenons, par exemple, le cas de figure où l'utilisateur sélectionne une entité à la souris [Figure 89]. Lors de la normalisation, le capteur console ne dispose que de l'adresse de l'entité sélectionnée. C'est donc le module de gestion de dialogue qui va, en premier lieu, affecter la valeur des traits « nom », « adjectifs » et « genre » à partir de la valeur des attributs de l'entité correspondants. C'est suivant ce même principe, et à l'aide de l'historique du dialogue, que sont complétés les traits d'une structure correspondant à un pronom, la recherche dans l'historique d'une structure compatible se faisant par tentative d'unification des traits. S'il est possible de trouver dans l'historique ou dans l'environnement, plusieurs structures susceptibles d'être unifiées, alors une erreur de compréhension est générée. C'est le cas, par exemple, où l'utilisateur énonce « détruit le panneau solaire » alors qu'il existe plusieurs panneaux dans l'environnement de travail.

Les erreurs de compréhension sont indiquées sous forme textuelle dans une widget, installée dans l'interface 2D de l'avatar par l'entité Constructeur. Nous avions tout d'abord envisagé un retour sonore en cas d'erreur de compréhension. Cette solution a cependant été abandonnée car elle a été jugée peu agréable par les utilisateurs.

5.2.4.3 La résolution des références intermodales

Une fois les structures complétées, le module de gestion du dialogue peut procéder à la résolution des références intermodales, références d'une modalité vers une autre modalité.

L'énoncé oral « pose ça là », accompagné de deux gestes de désignation (deux clics souris, par exemple), constitue l'exemple traditionnel [Bolt80] de ce type de situation. Le pronom « ça » et l'adverbe « là » font référence aux deux gestes de désignation. Le problème consiste alors à associer ces deux déictiques aux événements utilisateur correspondants (en l'occurrence les deux gestes de désignation).

Le but de la fusion multimodale consiste donc à associer, à chaque structure complément, contenant un déictique, une structure générée à partir d'un autre stimulus. Pour ce faire, il est nécessaire d'utiliser plusieurs critères permettant d'attribuer une note de pertinence à chaque association possible, c'est-à-dire à chaque couple (déictique, désignation). Le processus de fusion pourra alors utiliser une association si sa note est suffisamment élevée. Dans le cas contraire, il faudra déterminer une autre association ou, le cas échéant, générer une erreur de compréhension.

Cinq critères de pertinence ont été identifiés au cours d'une réunion de travail dans le cadre de l'atelier multimodal des Journées Interfaces IHM'92 :

- La proximité temporelle
- La complémentarité logique
- La complétude des événements
- Les contextes
- L'incompatibilité des modalités

Nous n'utilisons pas ce dernier critère car il n'est jamais vérifié dans notre système. Nous allons maintenant présenter les critères retenus pour évaluer la pertinence d'une association.

Le critère de proximité temporelle

Deux événements (structures de traits) pourront être fusionnés s'ils ne sont pas éloignés l'un de l'autre de plus d'une distance temporelle donnée. Cette distance peut être fixe ou calculée dynamiquement suivant les capacités de réaction de l'utilisateur. Il faut en fait deux distances temporelles : une durée antérieure à la prononciation du déictique (δ_1) et une durée postérieure à la prononciation de ce déictique (δ_2). Prenons l'exemple d'un énoncé composé d'une expression orale associée à un geste de désignation effectué à la souris [Figure 90].

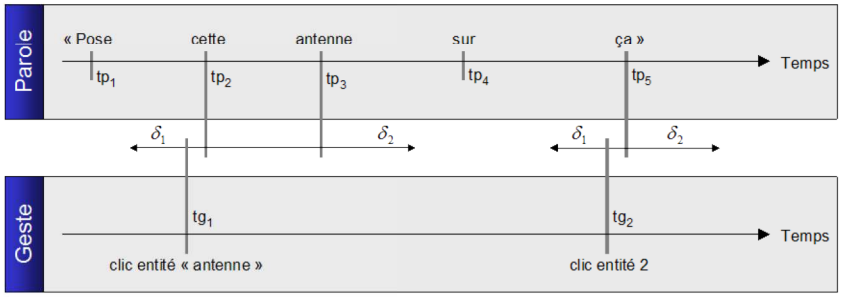


Figure 90 : *Critère de proximité temporelle*

Nous pouvons remarquer, sur cette figure, que le déictique « cette » a été prononcé à l'instant tp_2 et le déictique « ça » à l'instant tp_5. Le critère de proximité temporelle permet d'associer au mot « cette » l'événement souris 'clic entité « antenne »' ($tp_2-\delta_1 < tg_1 < tp_3+\delta_2$) et, au mot « ça », l'événement 'clic entité 2' ($tp_5-\delta_1 < tg_2 < tp_5+\delta_2$). Il est à noter que, « cette » étant un adjectif démonstratif, nous utilisons la date tp_3 de prononciation du mot suivant (« antenne ») pour déterminer la proximité temporelle postérieure de ce syntagme.

Le critère de complémentarité logique

Ce critère permet, dans certains cas, de fusionner des événements qui n'ont pas vérifié le critère de proximité temporelle. Ce type de situation peut survenir lorsque l'utilisateur commet une erreur en exprimant son énoncé.

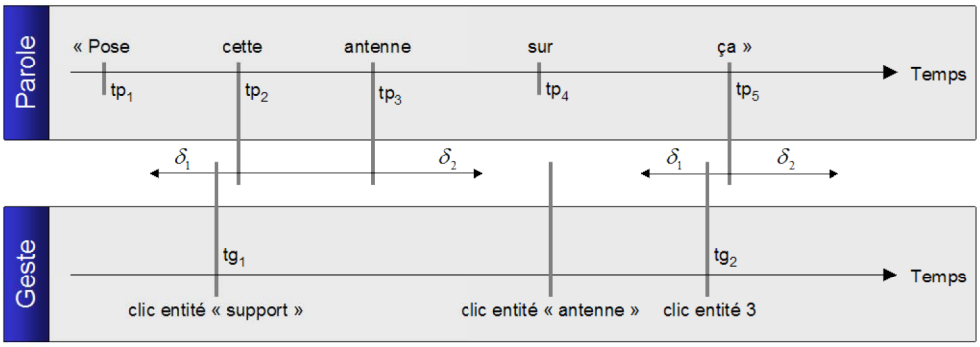


Figure 91 : *Critère de complémentarité logique*

Reprenons le cas de figure présenté précédemment. L'utilisateur clique maintenant par erreur sur le support du satellite qui se trouve près de l'antenne. Se rendant compte de son erreur, il clique immédiatement sur l'antenne qu'il veut déplacer. Dans ce cas le critère de proximité temporelle n'est vérifié que pour les couples (« cette antenne », 'clic entité « support »') et (« ça », 'clic entité 3'). Par contre, le critère de complémentarité logique est vérifié pour le couple (« cette antenne », 'clic entité « antenne »'). C'est donc ce dernier qui sera utilisé lors de la fusion multimodale.

Le critère de complémentarité logique peut porter sur un attribut quelconque de l'entité désignée, son nom, sa fonction, sa couleur. L'utilisateur peut ainsi énoncer des commandes du type « texture la batterie comme ça » où « ça » référence une entité. La vérification portera alors sur le fait que l'entité référencée possède bien une texture.

Les critères de priorité

Ces critères ne sont pas indispensables mais ils permettent néanmoins de 'départager' deux associations qui auraient obtenu la même note.

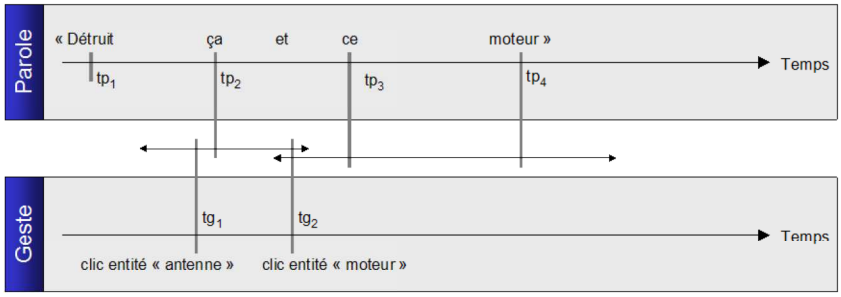


Figure 92 : *Critères de priorité par adjectif et par nom*

Prenons pour exemple le cas où l'utilisateur énonce oralement « détruit ça et ce moteur » en cliquant tout d'abord sur une antenne puis sur un moteur. Dans le cas de figure présenté [Figure 92], les couples (« ça », 'clic entité antenne') et (« ça », 'clic entité moteur') vérifient tous deux les critères de proximité temporelle et de complémentarité logique. Pour lever cette ambiguïté, nous utilisons un critère de priorité qui classe les couples entre eux. Ainsi, les couples (« ça », 'clic entité antenne') et (« ça », 'clic entité moteur') obtiennent la même note de priorité mais le couple (« ce moteur », 'clic entité moteur') obtient une note supérieure. La fusion aura donc lieu entre les couples (« ça », 'clic entité antenne') et (« ce moteur », 'clic entité moteur').

Le critère d'ordonnancement des références

Ce critère est vérifié si l'ordre temporel des syntagmes contenant des déictiques (références intermodales) est identique à l'ordre des entités désignées pour correspondre à ces déictiques. Il n'est pas vérifié si, par exemple, l'utilisateur énonce « accroche cet élément sur ce panneau » en désignant d'abord le panneau puis, ensuite, l'élément en question.

5.2.4.4 La fusion multimodale

C'est lors de cette étape qu'est générée la commande finale qui sera ensuite exécutée par le module interprète. Pour cela, les associations ayant les degrés de cohérence les plus élevés sont fusionnées pour ne produire, au final, qu'une seule structure de traits. Prenons par exemple le cas de figure où l'utilisateur énonce oralement « pose cette batterie et ça sur le socle » tout en cliquant sur deux entités.

Le module de gestion du dialogue reçoit alors du capteur la structure présentée [Figure 93].

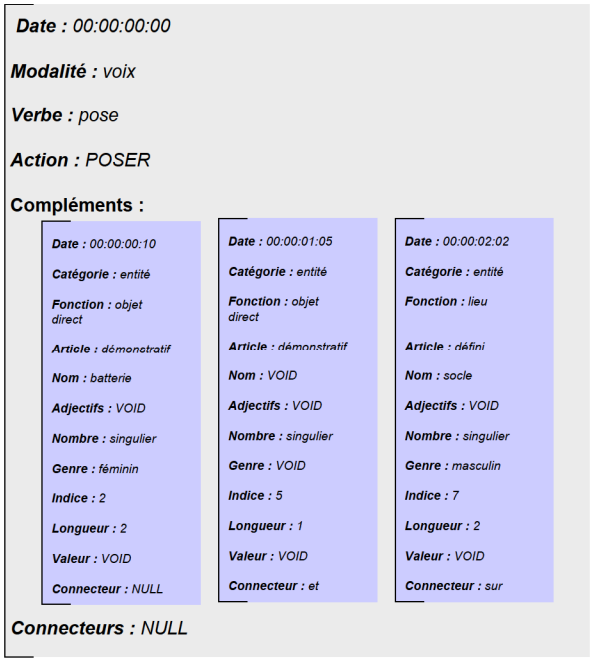


Figure 93 : *Structure issue d'un stimulus texte*

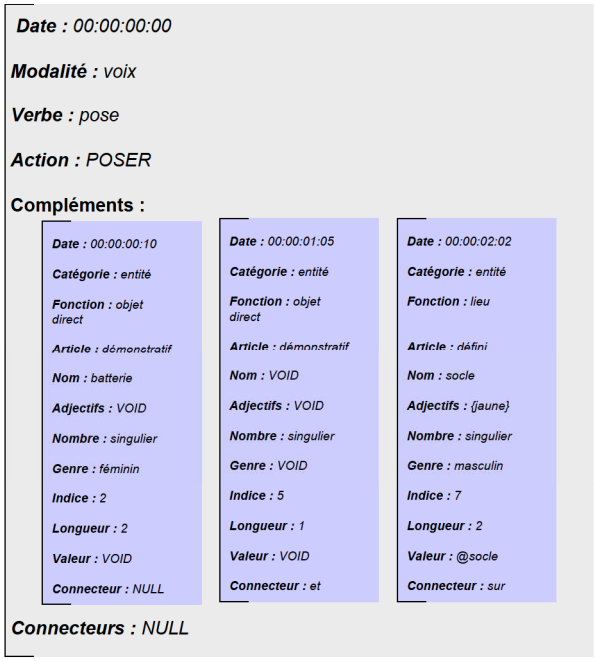


Figure 94 : *Structure après complétion*

On peut voir [Figure 94] que seuls les traits du complément numéro 3 (« le socle ») ont pu être complétés par unification. Le module de gestion du dialogue ne peut pas encore déterminer quelles sont les entités correspondant aux deux autres compléments (« cette batterie » et « ça »). Il doit, pour cela, utiliser les structures ([Figure 95] [Figure 96]) générées par le capteur d'ordres console à partir de stimuli souris.

Figure 95 : *Structure issue d'un stimulus souris*

Figure 96 : *Structure suivante issue d'un stimulus souris*

Il est alors possible, grâce au mécanisme de résolution des références intermodales, de fusionner les trois événements reçus en une seule structure complète [Figure 97]. La commande générée sera alors : POSER (@batterie, @émetteur, @support) ;

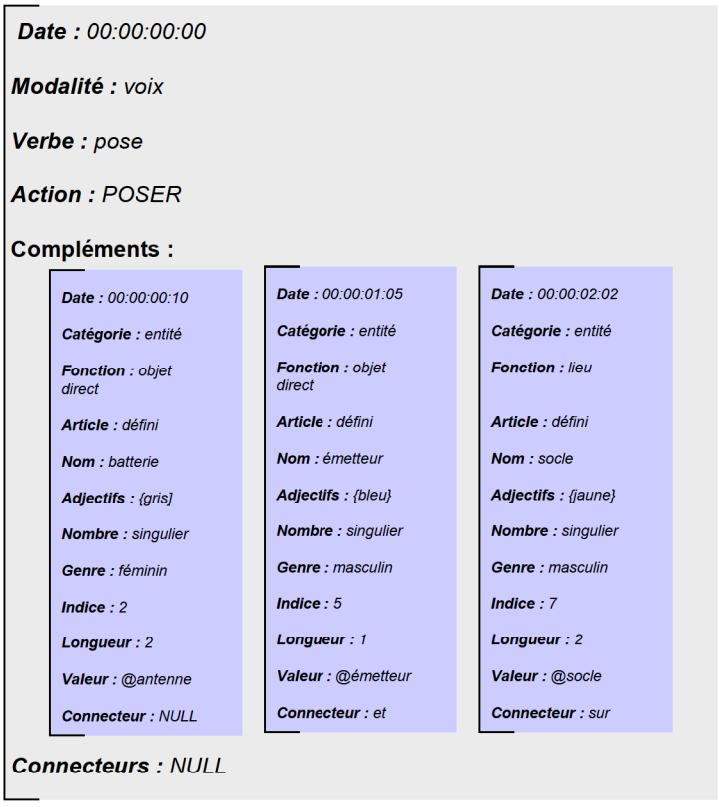


Figure 97 : *Structure complétée par fusion multimodale*

5.2.5 Le module interprète

Le rôle du module interprète est de rendre effectives les commandes produites [Balet94] par le module de gestion du dialogue. Ce module est également en charge de la traduction des paramètres dits 'flous' [Dubois97] en valeurs exactes. Une commande du type « déplace ce module vers la droite » sera, par exemple, interprétée comme une translation d'une direction horizontale vers la droite par rapport au point de vue. L'amplitude de cette translation sera proportionnelle à la taille de l'objet que l'utilisateur désire déplacer.

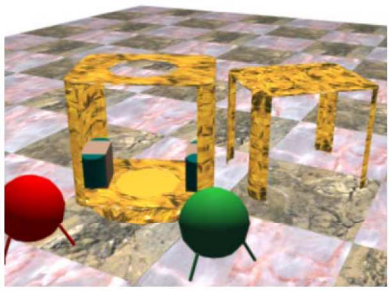


Figure 98 : *« déplace le réservoir bleu vers la droite »*

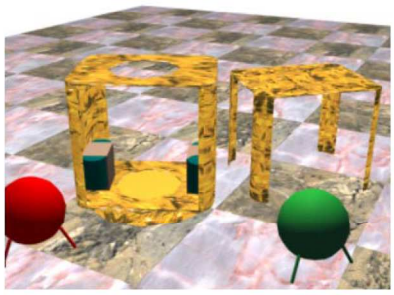


Figure 99 : *« déplace le vers l'arrière »*

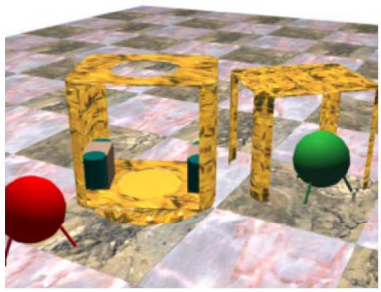


Figure 100 : *« encore »*

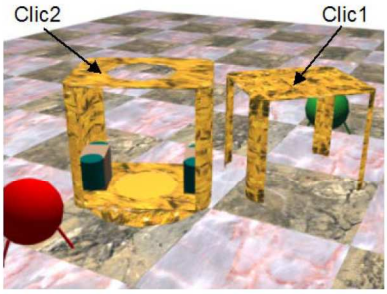


Figure 101 : *« aligne ça sur ça »*

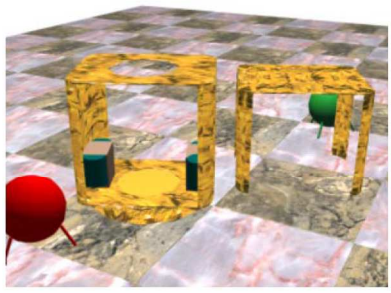


Figure 102 : *« colorie le réservoir rouge en vert et le bleu en blanc »*

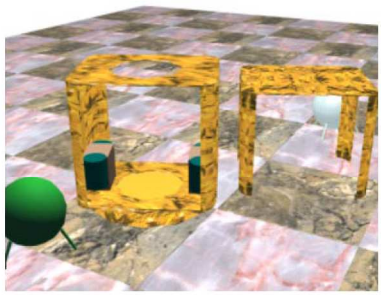


Figure 103 : *« supprime les »*

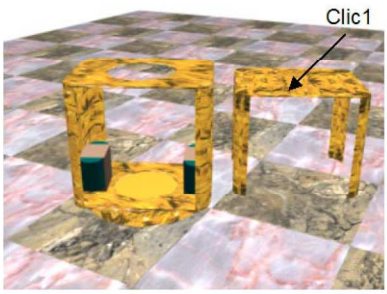


Figure 104 : *« pose ça sur le châssis »*

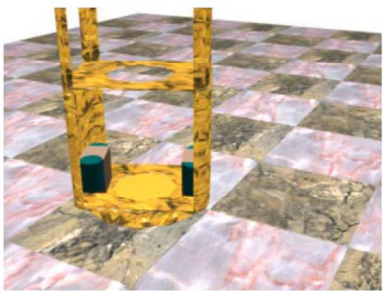


Figure 105 : *« annule deux fois »*

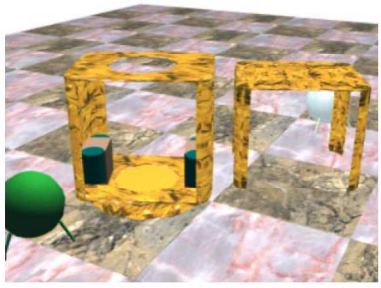


Figure 106 : *Résultat final*

5.3 L'Expert

Cette entité est en charge de vérifier la validité et la cohérence du prototype construit en fonction des règles métier du domaine d'application. Elle est donc supposée contenir la connaissance d'un expert de ce domaine. En pratique, nous nous sommes limités à l'implantation de fonctionnalités permettant simplement de mettre en lumière les concepts proposés.

L'entité Expert de PROVIS se contente juste de vérifier que le prototype construit ne dépasse pas un certain poids et que la consommation électrique n'excède pas la capacité des générateurs et autres systèmes d'alimentation en énergie utilisés. L'implantation d'un comportement plus élaboré pourrait être réalisée de manière à vérifier, par exemple dans le domaine du bâtiment, que toutes les règles de construction sont bien respectées.

Cette entité, comme de nombreuses autres, installe des widgets dans l'interface 2D des avatars. Ces éléments d'interface permettent de paramétrer l'ensemble des contraintes dont elle dispose.

Les résultats de l'analyse de l'entité Expert peuvent être communiqués sous forme textuelle, dans une boite de dialogue, ou sous forme sonore ou vocale. La bibliothèque MBROLA v2.05 [Dutoit97] pour la synthèse de la parole est à ce sujet en cours d'évaluation.

5.4 Les prototypes

Les entités Prototypes sont en fait les éléments géométriques que manipulent les utilisateurs. On parlera indifféremment, par la suite, de composants ou d'objets pour désigner les briques de base d'un prototype. Il n'existe conceptuellement aucune différence entre un prototype et un composant.

Un prototype [Figure 107] est défini par une forme géométrique au sens VRML du terme, c'est à dire un ensemble de géométries correspondant à ses différents niveaux de détails. La forme d'un prototype est transmise au système via un effecteur de forme 3D. Ce type d'entité est référencé par une adresse URL et dispose d'un attribut documentation ainsi que d'un profil de protection utilisé pour la gestion des droits d'accès.

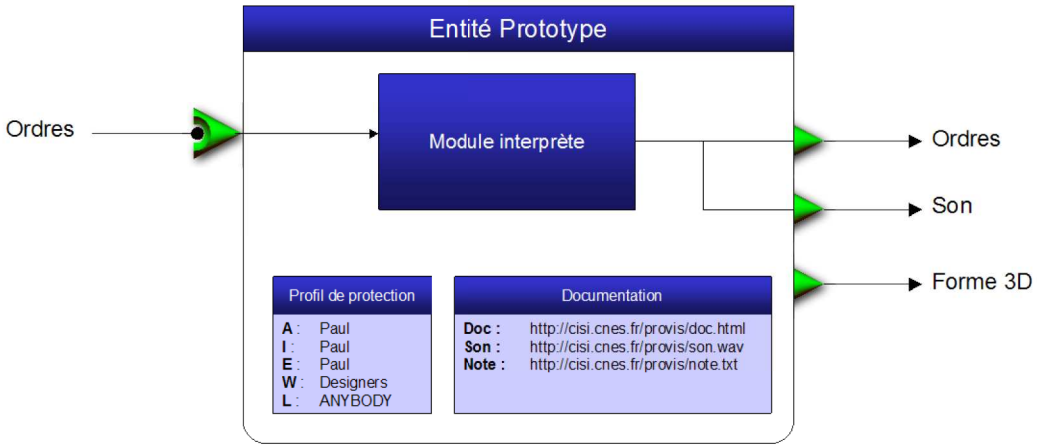


Figure 107 : *Architecture générale d'une entité prototype*

5.4.1 Droits d'accès

Chaque prototype dispose d'un profil de protection composé de cinq droits d'accès [Kanawati97], chacun d'eux étant attribué à une liste d'utilisateurs ou de groupe d'utilisateurs [Figure 107].

Administrer (A) : ce droit permet de modifier le profil de protection de l'entité prototype.

Importer (I) : ce droit permet d'importer une entité prototype dans l'environnement.

Exporter (E) : ce droit autorise l'exportation du prototype en vue, par exemple, d'une sauvegarde locale.

Editer (W) : ce droit permet de modifier le prototype.

Lire (R) : ce droit permet de voir l'entité prototype. Un utilisateur n'ayant pas le droit de lecture pour une entité ne verra que la boite englobante de celle-ci.

On peut, par exemple, restreindre la consultation du prototype complet à un sous-ensemble de ses composants pour un premier groupe d'utilisateurs et à un autre sous-ensemble pour un deuxième groupe.

Le profil de protection est, de plus, utilisé pour la gestion des accès concurrentiels. Par exemple, lorsqu'un utilisateur autorisé clique sur une entité prototype alors, celle-ci, de manière restrictive mais temporaire, lui attribue son droit d'écriture. Il sera restitué à sa valeur initiale lorsque l'utilisateur désélectionnera l'entité. Il est à noter qu'une entité sélectionnée change de forme pour s'entourer d'une boite englobante de la couleur correspondant à l'utilisateur l'ayant sélectionnée.

5.4.2 Documentation

Il est possible d'associer toutes sortes d'informations non graphiques à un prototype, aussi bien des notes orales que des fichiers au format HTML. Pour cela, l'entité prototype dispose d'une liste d'URLs regroupées sous son attribut documentation. Une entité avatar autorisée peut alors spécifier ou utiliser cette liste pour associer ou consulter de la documentation en ligne.

5.4.3 Assemblage de prototypes

Nous utilisons des liaisons cinématiques pour assembler plusieurs, généralement deux, composants entre eux. Elles permettent à l'utilisateur, dans un deuxième temps, d'interagir de manière réaliste avec l'assemblage tout en maintenant la cohérence de ses mouvements.

Une telle liaison sera par exemple créée dès lors que deux objets entreront en contact (lorsque l'utilisateur pose une batterie sur un support, une liaison reliant ces deux objets est créée). Nous proposons, pour ce faire, un modèle qui servira à définir le mouvement des objets avec ses contraintes et son influence sur l'environnement virtuel (un réservoir ne pourra par exemple pas traverser le socle sur lequel il est posé. De même, le mouvement de ce socle entraînera le mouvement de ce qu'il supporte).

5.4.4 Norme utilisée pour l'assemblage

La liaison unissant deux objets est choisie dans une bibliothèque de liaisons types (extraites de la norme AFNOR EO4-EO15 [Tableau 5]) ou de liaisons composées qui permettent de modéliser des mécanismes composés et donc plus complexes.

NB : Le Tableau 5 donne, pour chaque liaison type, le nombre de degrés de liberté (d.d.l.) ainsi que leur décomposition en rotation et translation [Chevalier69]. L'utilisation de cette norme revêt deux intérêts essentiels :

La simplicité de mise en œuvre : il est en effet très simple de gérer les contraintes de mouvements imposées par chacune de ces liaisons. De plus, à l'aide d'une méthode décrite au paragraphe 5.4.6, il est possible d'augmenter cette liste de liaisons en les combinant entre elles pour former des liaisons composées.

L'évolution vers une interaction plus réaliste : ne disposant pas de systèmes à retour d'efforts performants, nous n'avons pas, pour l'heure, intégré certaines notions de mécanique telles que les propriétés physiques des liaisons ou les frottements. Il est à noter, toutefois, que de telles extensions sont possibles à partir de la norme utilisée [Dumont 90].

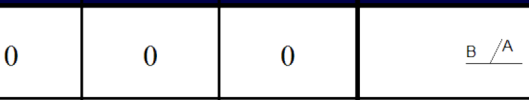
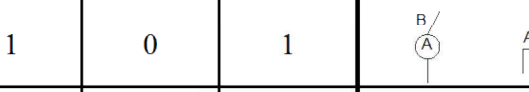
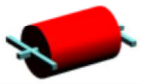
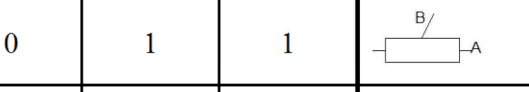
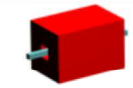
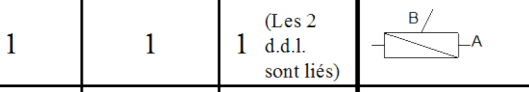
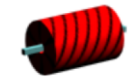
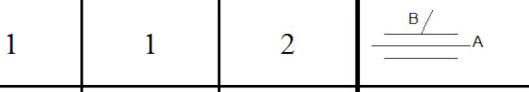
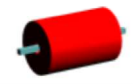
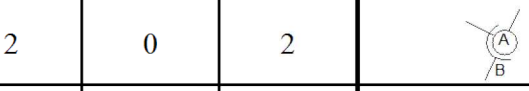
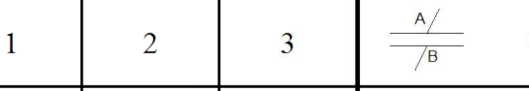
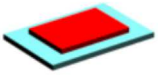
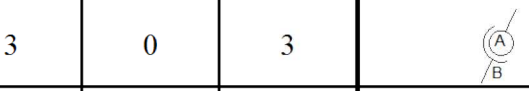
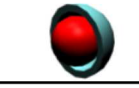
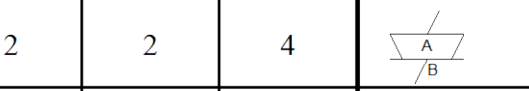
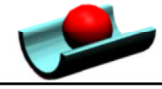
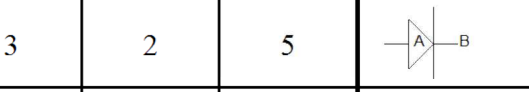
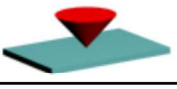
Liaison	Rotation	Translation	Nb. de d.d.l.	Symbole plan	Icône 3D
Encastrement ou Fixe	0	0	0		
Pivot	1	0	1		
Glissière	0	1	1		
Hélicoïdale	1	1	1 (Les 2 d.d.l. sont liés)		
Pivot glissant	1	1	2		
Sphérique à doigt	2	0	2		
Appui plan	1	2	3		
Rotule ou Sphérique	3	0	3		
Linéaire rectiligne	2	2	4		
Linéaire annulaire	3	1	4		
Ponctuelle	3	2	5		
Libre	3	3	6	Pas de symbole	

Tableau 5 : *Liaisons cinématiques normalisées*

5.4.5 Liaisons cinématiques simples

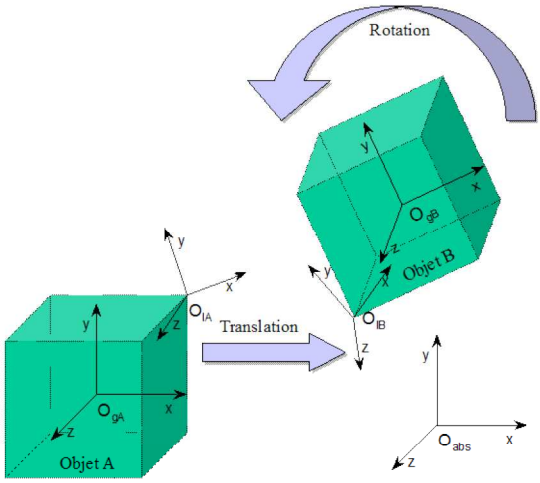


Figure 108 : *Liaisons cinématique reliant deux composants*

5.4.5.1 Modèle utilisé

La position dans l'espace de chacune des entités est définie à l'aide d'une matrice exprimant la transformation entre le repère absolu et le repère de leur centre d'inertie (noté respectivement O_{gA} ou O_{gB} dans la Figure 108).

L'association de deux objets par une liaison type introduit, de plus, deux repères supplémentaires (R_{lA} et R_{lB} de centres respectifs O_{lA} et O_{lB} qui sont les points d'ancrage de la liaison sur les objets A et B). Ces repères, qui seront appelés par la suite repères de liaison, déterminent, pour chacun des deux objets, la position et l'orientation de la liaison dans leur repère d'inertie respectif. La définition d'une liaison contient alors, pour chacun des objets, la matrice de passage de son repère de liaison à son repère d'inertie. Il est à noter que, pour un objet donné, cette matrice est constante car le point d'ancrage de l'objet reste fixe par rapport à son centre d'inertie.

Ainsi, le mouvement entre les deux repères de liaison des objets détermine sans ambiguïté leur mouvement relatif.

Une liaison type sera alors caractérisée par :

- Les deux objets reliés et leur matrice de liaison respective.
- La matrice *Ml* définissant le type de liaison : cette matrice représente la matrice de passage du repère R_{lB} de liaison d'un objet B au repère R_{lA} de liaison d'un objet A. Chaque liaison de la norme AFNOR peut être définie par une telle matrice [Arnaldi88] qui est utilisée, dans notre système, pour décrire les mouvements autorisés entre deux objets.

- La matrice *Mc* définissant les contraintes de mouvement sur la liaison. Cette matrice représente les limites de variation pour chacun des degrés de liberté. Elle est par exemple de la forme, pour une liaison de type pivot :

$$\begin{pmatrix} \text{min} & \text{max} & \\ 0 & 0 & \rightarrow \text{Translation suivant } \vec{x} \\ 0 & 0 & \rightarrow \text{Translation suivant } \vec{y} \\ 0 & 0 & \rightarrow \text{Translation suivant } \vec{z} \\ 0 & 90 & \rightarrow \text{Rotation autour de } \vec{x} \\ 0 & 0 & \rightarrow \text{Rotation autour de } \vec{y} \\ 0 & 0 & \rightarrow \text{Rotation autour de } \vec{z} \end{pmatrix}$$

Malgré sa simplicité, une telle matrice permet de modéliser de manière tout à fait satisfaisante les contraintes des liaisons utilisées pour nos travaux.

5.4.5.2 Quelques exemples de matrices de liaison

Liaison encastrement : pour ce type de liaison, *Ml* est la matrice unité car il ne peut y avoir de mouvement relatif entre les deux objets dont les repères de liaison coïncident.

$$Ml = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Liaison pivot : le seul mouvement relatif possible est une rotation d'un objet par rapport à l'autre autour d'un axe (ici l'axe des z),

$$Ml = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

où α représente l'angle de rotation autour de l'axe.

Liaison glissière : le seul mouvement autorisé est la translation *t* suivant un axe (ici l'axe des z).

$$Ml = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & t \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Liaison hélicoïdale : la liaison hélicoïdale est la combinaison d'une liaison glissière et d'une liaison pivot à ceci prés que la rotation et le glissement sont liés. Si l'on note *pas* la longueur parcourue en translation par l'objet B lorsque celui-ci effectue un tour complet sur lui même, alors pour une rotation de α degrés, l'objet B effectue une translation de $\alpha.pas/2\pi$.

$$Ml = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & \frac{\alpha \times pas}{2\pi} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Liaison pivot glissant : cette liaison est, comme la précédente, combinaison d'une liaison glissière et d'une liaison pivot. Cependant les degrés de liberté en translation, noté t, et en rotation, noté α, sont ici indépendants.

$$Ml = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & t \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Liaison appui plan : cette liaison possède un degré de liberté en rotation noté α et deux degrés en translation, notés t1 et t2.

$$Ml = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & t1 \\ \sin\alpha & \cos\alpha & 0 & t2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

5.4.5.3 Mise en place du modèle

Lorsque l'opérateur décide de mettre en mouvement une entité, il peut utiliser les commandes appropriées de l'interface, un énoncé en langage naturel fourni à l'entité Constructeur ou une commande manuelle saisie à l'aide du contrôleur 3D. Dans ce dernier cas, l'opération effectuée sera traduite sous la forme d'une matrice exprimée dans le repère local de l'entité concernée. Il appartient ensuite à cette dernière de déterminer si la commande peut être exécutée entièrement ou partiellement en fonction du type des liaisons (et des contraintes associées) qu'elle possède.

Prenons pour exemple une scène constituée de deux objets :

- Un rail R
- Une locomotive L.

L est posé sur R de telle sorte qu'une liaison de type glissière unisse les deux. De plus, on n'autorise que le mouvement de translation selon l'axe directeur de R.

Si l'opérateur prend virtuellement L et lui donne un mouvement respectant la liaison (la matrice correspondant au mouvement désiré est donc 'compatible' avec les matrices Ml et Mc de L) alors la locomotive, par le jeu de multiplications de matrices, suivra la direction indiquée.

Si maintenant l'opérateur fait la même expérience en imposant un mouvement ne respectant pas complètement Ml (par exemple, une translation de vecteur non parallèle à R) alors, deux comportements peuvent être envisagés :

- Ou bien l'entité L ignore la commande en prévenant l'utilisateur qu'il a fait une fausse manœuvre.
- Ou bien elle ne considère que la partie valide du mouvement (dans l'exemple la composante en x).

Cette dernière solution est vraisemblablement la meilleure car, alors qu'il est possible à l'aide des commandes de l'interface de diriger précisément un objet, l'absence de retour d'effort sur le périphérique utilisé interdit encore les manipulations très précises. Il est à noter que le retour tactile par génération de vibrations suffirait pour ce type d'opération.

5.4.6 Liaisons cinématiques composées

Nous venons de décrire un modèle de liaison simple qui permet théoriquement la construction de tous les types de mécanismes. Cependant, ce modèle reste inadapté pour simuler des liaisons complexes et leur manipulation dans des applications à forte contrainte temps réel.

Prenons par exemple le mécanisme d'une targette [Figure 109]. Ce dernier pourrait être modélisé simplement à l'aide d'une liaison de type pivot glissant, la gestion des collisions entre l'élément mobile et son support assurant le respect des contraintes et l'intégrité du mouvement. Bien que d'aspect séduisant, cette approche est néanmoins beaucoup trop grande consommatrice de puissance de calcul. A chaque mouvement de la partie mobile, l'ordinateur devrait tester si un des polygones qui la compose entre en contact avec ceux définissant la forme du socle. Cette méthode est d'autant moins appropriée qu'il est possible d'utiliser un certain nombre d'informations, définissant le mécanisme en question, afin de s'affranchir de tout test de collision. Nous utilisons, pour cela, le modèle de liaison composée que nous proposons dans ce paragraphe.

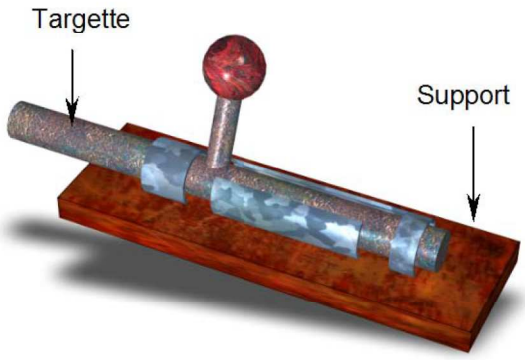


Figure 109 : *Un exemple de mécanisme composé, la targette*

Reprenons donc l'exemple choisi. Ce dernier peut être décomposé en trois situations, chacune d'elles étant caractérisée par une liaison type :

Situation A : dans cette position [Figure 110], la liaison qui unit la targette et son support est une liaison de type pivot. En effet, le mouvement de la targette relativement à son support est ici limité à une rotation autour de son axe. De plus, l'angle de rotation doit être contraint entre les valeurs 0° et 180° (cet angle étant mesuré par rapport à la position d'origine de la targette).

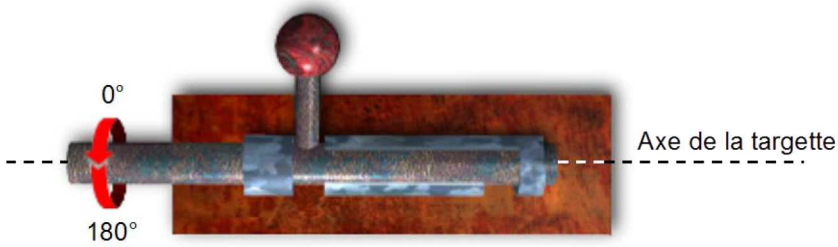


Figure 110 : *Mécanisme de la targette en situation A*

Situation B : dans cette position [Figure 111], la liaison caractéristique est une liaison glissière, le mouvement de la targette étant limité à une translation le long de son axe. La contrainte imposée à la translation est qu'elle soit bornée entre 0 et longueur, cette mesure étant effectuée relativement à la position d'origine de la targette sur son axe.

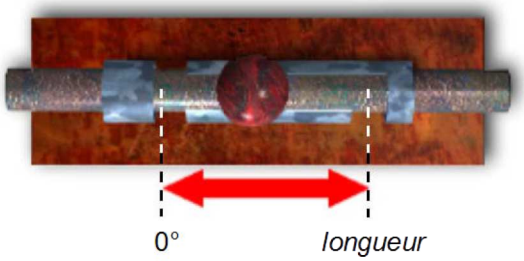


Figure 111 : *Mécanisme de la targette en situation B*

Situation C : cette situation [Figure 112] est caractérisée par une liaison pivot qui est identique à celle de la situation A à ceci près que l'angle de rotation est maintenant contraint entre 0 et 90° (cet angle est mesuré par rapport à la position d'origine - i.e. d'arrivée - de la targette dans cette situation).

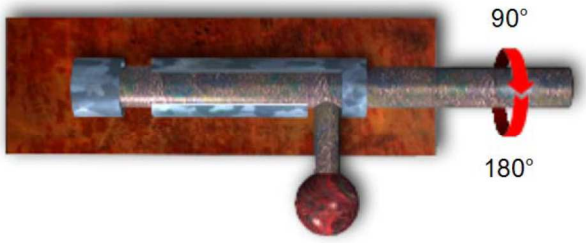


Figure 112 : *Mécanisme de la targette en situation C*

Une seule des trois situations étant effective à la fois, l'utilisateur agit sur la targette pour la faire passer de l'une à l'autre. Ce passage obéit bien évidemment à certaines contraintes :

Passage de A vers B : la targette pourra passer de la situation A à la situation B à la seule condition que l'angle de rotation sur la liaison pivot soit égal à 90° (on pourra cependant tolérer un certain jeu dans le mécanisme en imposant un intervalle, par exemple [85°, …, 95°], plutôt qu'une valeur unique).

Passage de B vers A : le passage de la situation B à la situation A pourra être effectué à condition que la translation relative de la targette sur son axe soit nulle, en d'autres termes, à condition que la targette soit en butée gauche sur son axe.

Passage de B vers C : le passage de la situation B à la situation C sera possible à condition que la translation effectuée par la targette sur son axe soit égale à longueur (i.e. la targette est alors en butée droite sur son axe).

Passage de C vers B : la targette pourra passer de la situation B à la situation C à condition que l'angle de rotation sur la liaison pivot de la situation C soit nul.

Il est à noter que, comme pour le passage de la situation A à la situation B, il est toujours possible, à l'aide d'intervalles, de donner des contraintes de passage moins strictes.

5.4.6.1 Modèle proposé

Comme nous l'avons vu, une liaison composée est constituée de plusieurs liaisons types qui définissent des situations (des états). De plus, le passage de l'un à l'autre de ces états obéit à certaines conditions. Il s'agit bien là de définir le comportement d'une entité au sens défini chapitre 5.2.6. Le comportement d'une entité disposant d'une « liaison composée » sera donc décrit par un Réseau de Petri à Objets où les jetons qui circulent sont des objets de type « liaison simple ».

5.4.6.2 Mise en place du modèle

Comme nous l'avons vu précédemment, une liaison composée est constituée de plusieurs liaisons types, chacune d'elles définissant une situation. Nous représentons chaque situation par une place dans le réseau de Petri à objets associé. Les arcs du réseau représentent alors le passage d'une situation à une autre. Enfin, les contraintes de passage sont placées dans la partie précondition de la transition associée à un arc. La description du Réseau de Petri à Objets associé au mécanisme de la targette est donné Figure 113.

Soit *m* le mouvement que l'opérateur désire imposer à la targette, l'objet m étant instance de la classe VrMouvement décrite de manière très simplifiée ci-dessous, nous nous limiterons dans cet exemple à des mouvements composés d'une translation et/ou d'une rotation suivant l'axe des x (i.e. l'axe de translation de la targette). Le mouvement *m* est transmis de l'entité avatar à l'entité manipulée via un stimuli de type console.

```
Class VrMouvement
Begin
Public :
   Int          mTransX;
   VrAngle      mRotX;
End;
```

Soit *l* un objet instance de la classe VrLiaison (spécifiée ci-après dans sa version simplifiée) qui représente la liaison simple courante passant de place en place suivant les mouvements effectués.

```
Class VrLiaison
Begin
   Int          x;
   VrAngle      rotx;
Public :
   VrObjet      support;
   VrObjet      o;

   /***** Le mouvement est il compatible ? ******/
   Boolean      Compatible( VrMouvement );
   /**** Appliquer un mouvement à la liaison ****/
   Void         Appliquer( VrMouvement );
End;
```

Soient *lA*, *lB* et *lC* les liaisons respectivement associées aux situations A, B et C. Il est à noter que les contraintes de passage d'une situation à une autre sont strictes dans cet exemple afin d'assurer le déterminisme du réseau associé au mécanisme de l'exemple. En effet un mouvement ne peut être ici compatible avec deux liaisons différentes afin de simplifier les pré-conditions présentées Figure 113.

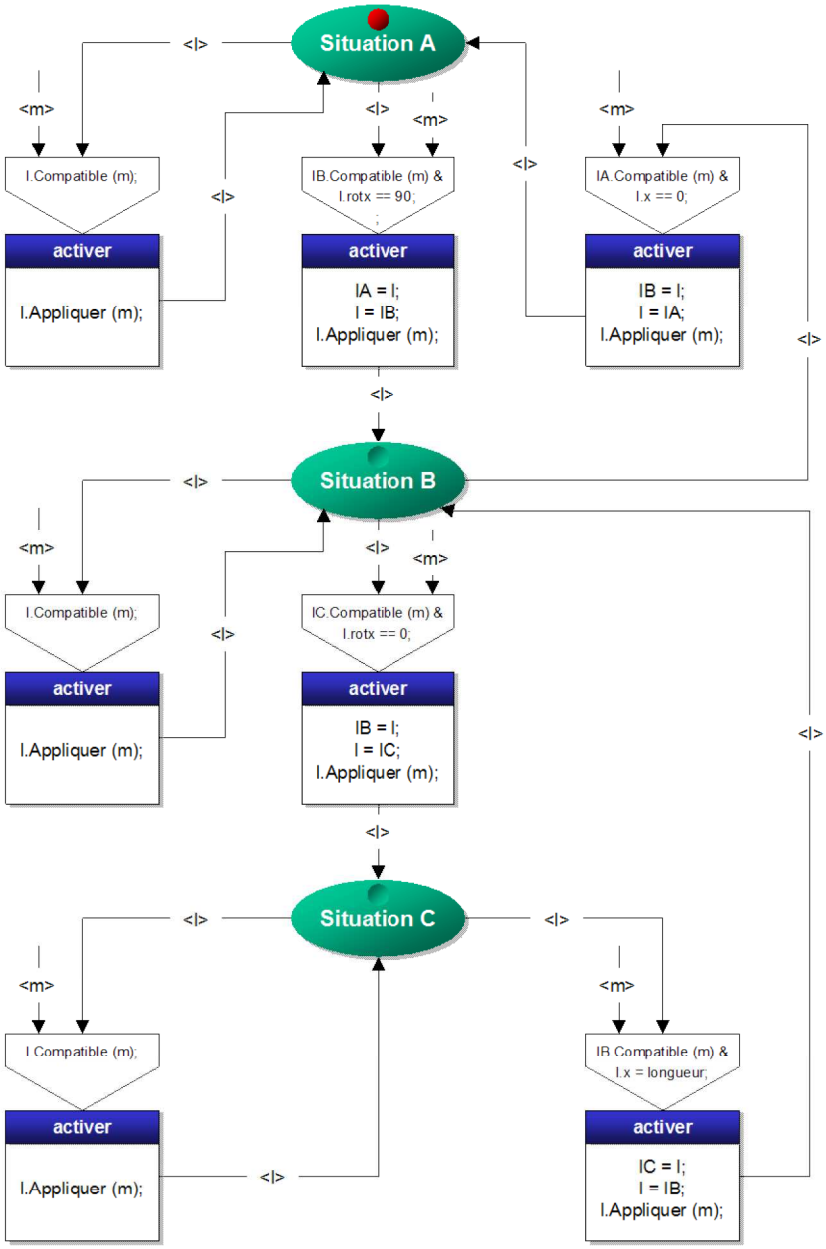


Figure 113 : *Modélisation du comportement du mécanisme de la targette par réseau de Petri à objets*

5.5 Conclusion

Ce chapitre a présenté le système PROVIS développé à l'aide de la plate-forme logicielle VIPER. Ce système, dédié au prototypage virtuel coopératif, tire pleinement profit des mécanismes de distribution, de partage des données et de télécommunication proposés par VIPER. Nous avons ainsi décrit comment, en utilisant quatre types d'entités différents, il est possible de définir un tel système.

Nous avons tout d'abord présenté les entités avatar qui sont en fait les clones numériques des utilisateurs. Nous avons justifié leur existence en montrant comment elles permettaient d'homogénéiser l'architecture de notre système et notamment la communication entre un utilisateur et les autres entités. Nous avons ensuite détaillé les différents modules comportementaux qui définissent les fonctionnalités d'une telle entité, à savoir :

- son module de visualisation lui permettant de produire une vue tridimensionnelle de l'environnement de travail,
- son module de gestion des communications qui permet l'administration des différents canaux de communication avec les utilisateurs distants,
- son module de reconnaissance vocale, en charge de la traduction d'un énoncé oral sous une forme textuelle,
- son interface 2D, lui permettant de recevoir les éléments d'interface nécessaires au dialogue avec les autres entités.

Nous avons de plus présenté les différents modes d'interaction proposés par l'entité avatar ainsi que leur évaluation, en terme d'ergonomie, par rapport aux solutions traditionnelles.

Nous avons ensuite introduit l'entité Constructeur qui défini fonctionnellement le système PROVIS. Ce faisant, nous avons tout d'abord détaillé le principe de normalisation des stimuli que nous proposons pour permettre la gestion du dialogue multimodal. Nous avons, de plus, présenté les deux modules comportementaux qui composent l'entité Constructeur :

- le module de gestion du dialogue, en charge de construire, par fusion multimodale, les commandes à exécuter,
- le module d'interprétation qui a pour objectif d'exécuter ces commandes.

Nous avons poursuivi en décrivant brièvement l'entité Expert qui a pour tâche de vérifier la validité du prototype virtuel en cours de construction à partir de règles métier. Cette entité fera sans aucun doute l'objet de futures recherches qui permettront de développer les concepts proposés.

Enfin, nous avons décrit les entités prototypes qui sont les éléments graphiques de l'environnement de travail. Comme nous l'avons vu, ces entités possèdent des attributs de documentation permettant au concepteur de leur attacher des fichiers de différentes sortes (HTML, son, etc.). C'est ainsi que PROVIS peut servir de support à la documentation en ligne. Elles possèdent en outre des attributs permettant la gestion des droits d'accès en vue de la protection des travaux réalisés. Pour finir, nous avons proposé un modèle de liaison composée pour l'assemblage de deux composants. Ce modèle permet ainsi de décrire et de gérer des contraintes de mouvement complexes, lors de la manipulation interactive de prototypes, sans avoir recours à une méthode de détection des collisions.

PROVIS n'est pour l'heure qu'un prototype de produit en cours d'industrialisation par CISI. Cependant, il est déjà utilisé en interne pour la construction de maquettes numériques (station Alpha [Figure 114]) ce qui nous a permis de valider les choix proposés et de confirmer son haut degré d'interactivité. Par là même, nous avons aussi pu identifier de nouveaux besoins quant à l'interaction avec des prototypes articulés, en l'occurrence des câbles, ou la construction d'entités graphiques qui pourraient évoluer, de manière autonome, au sein de l'environnement de travail.

Figure 114 : *Intérieur de la future station spatiale internationale*

6 Simulation comportementale

Au cours du développement de notre projet, il nous est apparu important de pouvoir doter certaines entités d'une autonomie de comportement. Par exemple, il peut être utile pour un concepteur ou un ingénieur de vérifier l'accessibilité d'un composant du prototype virtuel sur lequel il travaille. Une méthode simple pourrait consister à proposer une entité de type « mannequin » que l'utilisateur manipulerait directement afin de déterminer si, par exemple, d'une position donnée l'entité peut atteindre un objet ou une autre position. Ce problème s'apparente à celui de l'animation de personnages et plus généralement de l'animation de systèmes articulés. Il est cependant bien connu que l'animation directe constitue une méthode très fastidieuse puisqu'il faut définir les mouvements pour chaque partie articulée. Certes, il est possible de simplifier la tâche de l'utilisateur en proposant des algorithmes de cinématique ou de dynamique inverse. Dans ce cas, l'utilisateur ne se soucie plus que de déplacer l'extrémité des chaînes articulées, ce qui reste encore, dans le cas de l'animation d'un personnage, une opération longue et souvent compliquée.

Nous proposons de traiter ce type de problème en affectant un comportement autonome à l'entité articulée qui devra trouver elle-même la solution au problème défini par l'utilisateur. Pour cela, nous utilisons une approche, nommée simulation comportementale, qui se rapproche de la vie artificielle de par ses principes. Ces entités ne sont alors plus régies par un système global gérant le mouvement de toutes les entités mais par un mécanisme de décision local placé en chacune d'elles.

Pour ce faire, nous avons choisi l'approche évolutionniste pour la définition de comportements « intelligents » au sein du système PROVIS. Nous présenterons, dans le paragraphe suivant, les principes de base d'une telle approche. Nous poursuivrons ensuite en présentant, paragraphe 6.2, les

principales recherches dans le domaine de la génération de comportements pour l'animation d'entités articulées par approche évolutionniste. Enfin, nous détaillerons la solution que nous proposons pour permettre le déplacement autonome et la manipulation de telles entités en temps réel.

6.1 Les systèmes évolutionnistes

Les systèmes évolutionnistes constituent un nouveau type d'algorithme facilitant la résolution de problèmes. Ils ont été introduits parallèlement aux Etats-Unis [Holland75] (algorithmes génétiques) et en Allemagne [Rechenberg73] (programmation évolutionniste). Popularisés dès 1989 par Goldberg [Goldberg89], ces algorithmes s'appuient sur une même idée, essayer de copier les mécanismes d'évolution de la nature.

Leur création a pour origine une remise en cause, dans les années 1970, des fondements de l'intelligence artificielle qui postulait qu'en connaissance des règles de base qui régissent un problème, on pouvait nécessairement le résoudre par inférence sur ces règles et sur ses conditions initiales. Se rendant compte de la difficulté d'application de tels postulats, l'intelligence artificielle s'est alors tournée vers l'étude de la nature afin de trouver de nouvelles solutions pour la résolution de problèmes. Cette observation de l'existant a donné naissance aux réseaux de neurones (par imitation du fonctionnement du cerveau) et aux systèmes évolutionnistes. Ces derniers copient le mécanisme naturel d'évolution pour fournir la solution à un problème. Ils agissent donc sur une population de structures de données en y appliquant des opérations calquées sur la nature.

Les systèmes évolutionnistes fournissent un moyen de recherche de solutions performant (mais souvent coûteux en calculs) dans des environnements où l'analyse classique se révèle difficile à mettre en œuvre (environnements non linéaires, problèmes sous-contraints, environnements dynamiques). La recherche des solutions s'appuie en effet sur l'expérience acquise par les individus présents dans l'environnement qui, de par sa pression sélective, propage dans la population les caractères génétiques qui portent des germes de solutions. C'est donc l'environnement et la structure de données contenue dans les chromosomes artificiels (le codage des solutions) qui vont spécifier le problème.

6.1.1 L'évolution naturelle

Dans la nature, l'acide désoxyribonucléique (ADN) est le dépositaire du code génétique de presque tous les organismes vivants, à l'exception de quelques virus qui utilisent l'acide ribonucléique (ARN) comme matériel génétique. Ce code, aussi appelé génotype, permet de diriger le développement de l'individu vers sa forme visible, le phénotype. Le phénomène complexe de morphogenèse permet le passage d'une information relativement compacte et unique constituée d'enchaînements d'acides aminés à un individu complet constitué de millions de cellules différenciées. Une espèce est quant à elle constituée d'un ensemble de caractéristiques communes à plusieurs chaînes d'ADN. Les chromosomes humains ont ainsi un patrimoine commun qui va définir la qualité d'humain mais aussi des différences qui vont par exemple coder la couleur des yeux ou des cheveux.

Le patrimoine commun d'une espèce est défini par la pression qu'exerce le monde extérieur (ou environnement) sur ses individus. Les traits communs sont alors ceux qui ont été créés car ils étaient importants pour la survie de l'espèce. Les traits auxiliaires sont libres car ils sont négligeables par rapport à la pression de l'environnement.

La nature a donc trouvé un support unique pour représenter les formes de vie. Elle a de même fourni à ce support des moyens de se copier et de se modifier. Ces derniers pouvant être internes, comme dans le cas du croisement génétique qui va dupliquer une chaîne d'ADN, ou externes, comme la mutation intervenant sous le fait d'agents extérieurs en altérant la chaîne lors de cette même copie. Cet ensemble de mécanismes a permis, à partir d'une chaîne d'ADN primitive, de créer toutes les formes de vie de notre planète.

6.1.2 Modèle artificiel

Un système évolutionniste va donc faire évoluer une population d'individus, constitués d'un ou plusieurs chromosomes artificiels modélisés par des structures de données. Chaque structure (génotype) représente un essai de solution (phénotype) portée par un chromosome. Sous la pression de l'environnement, ces structures vont évoluer et converger vers au moins une des solutions au problème.

Un tel système peut donc être vu comme un ensemble d'opérations faisant converger une population vers la solution d'un problème. Le processus mis en œuvre peut être décomposé en une étape de création de la population initiale P_0 et trois étapes que l'on réitère jusqu'à l'obtention de la solution cherchée [Figure 115].

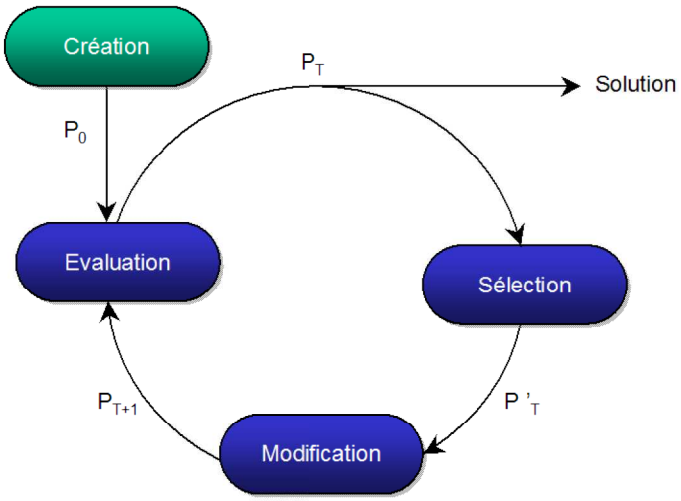


Figure 115 : *Le cycle principal d'un système évolutionniste*

6.1.2.1 La création

Cette opération permet de générer la population initiale. Elle est généralement réalisée de façon aléatoire tout en respectant certaines contraintes évidentes liées au problème. Par exemple, dans le cas de la recherche de solutions à une équation mathématique, on ne générera pas d'individus ayant une valeur négative si l'on cherche une solution dans $\Re^+$. La pertinence des choix faits lors de la création va en fait permettre d'augmenter la rapidité de convergence de l'algorithme en lui fournissant une population initiale plus homogène. Cette présélection risque néanmoins de supprimer des informations importantes dans le patrimoine génétique global de la population, empêchant ainsi l'algorithme d'atteindre le maximum global de sa fonction d'adaptation.

Il existe plusieurs modèles possibles pour représenter les individus de la population :

Chaînes de bits

Les individus sont représentés par des chaînes de bits de longueur fixe comme support du code génétique. Cette représentation a été choisie par Holland [Holland75] car elle permet un support simple des opérations évolutionnistes et est de plus applicable à un grand nombre de problèmes. On peut en effet coder simplement à l'aide d'une chaîne de bits l'ensemble des entiers et des réels, et donc optimiser un grand nombre de fonctions mathématiques.

Figure 116 : *Codage chromosomique sous forme de chaîne de bits*

Outre sa simplicité de mise en œuvre, cette méthode possède des fondements mathématiques permettant de justifier sa convergence. Cette preuve de convergence se base principalement sur la théorie des schémas [Holland75]. Cerf [Cerf94] a donné une autre preuve de convergence en s'appuyant sur une analyse par chaînes de Markov.

S_Expressions

Ce modèle, introduit par [Koza92], utilise des S_Expressions et non plus des chaînes de bits pour représenter les individus. Cette représentation sous forme d'arbres permet d'utiliser des structures non linéaires et se place donc en approche de plus haut niveau par rapport au codage par chaînes de bits.

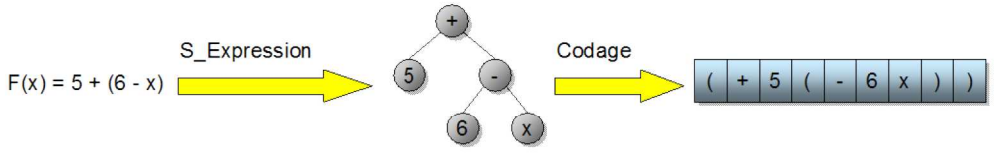


Figure 117 : *Codage chromosomique basé sur les S_Expressions*

Elle permet de plus une mise en œuvre simple de programmes évoluant dans des environnements dynamiques. Il est en effet beaucoup plus aisé de manipuler des mots de haut niveau permettant de créer de véritables programmes que de devoir décoder des chaînes binaires. S'appuyant sur une définition proche du langage LISP, elle a en outre bénéficié d'extensions fondées sur ce langage telles que la définition automatique de composantes de haut niveau (correspondant à des fonctions) à partir du vocabulaire de base [Koza94].

Il faut cependant remarquer que, même si cette technique a permis de résoudre une grande quantité de problèmes, ses fondements sont uniquement expérimentaux. Il n'existe en particulier aucune preuve théorique de sa convergence. De plus, la recherche d'un vocabulaire est le plus souvent effectuée de manière empirique et ne dépend donc que de l'expérience de l'utilisateur.

6.1.2.2 L'évaluation

Cette opération assigne une note d'adaptation, ou «fitness», à chaque individu de la population. Cette note détermine la qualité de la solution proposée par l'individu par rapport au problème posé. Son rôle est donc déterminant car c'est cette étape qui dirige l'évolution et permet d'adapter la pression de l'environnement. Cette fonction va donc associer à un point de l'espace d'état Ψ (c'est à dire un génotype) une valeur réelle. C'est donc une fonction $F : \Psi \rightarrow \Re$ qui associe, à une configuration du génotype, une note représentant l'adaptation du phénotype qu'il représente.

La fonction d'évaluation n'a que peu de contraintes puisqu'il suffit qu'elle puisse être rendue positive pour assurer une compatibilité avec la théorie des « schémas » qui constitue une approche de démonstration de la convergence du système [Goldberg89]. Elle ne subit donc aucune contrainte de type dérivabilité ou continuité ce qui simplifie grandement son écriture.

6.1.2.3 La sélection

Cette opération modifie la composition de la population afin de permettre aux individus, ayant le plus d'aptitudes face au problème, de propager leurs caractéristiques. Elle va donc servir à la régulation de la pression de l'environnement sur la population. Si cette fonction est trop exigeante (sélection unique des individus ayant la meilleure adaptation), l'évolution sera trop guidée et l'algorithme se contentera d'explorer des minimas locaux. Si elle est trop lâche, la convergence sera très lente voire inexistante. Plusieurs méthodes de sélection sont présentées dans [Luga97] afin de pallier ces problèmes.

6.1.2.4 La modification

Il existe plusieurs opérations de modification qui peuvent être appliquées aux chromosomes qui composent la population. Dans le cas où les individus sont représentés sous forme de S_Expressions, les positions de modification sont choisies de manière à produire un effet conservant la cohérence de la représentation.

Le croisement : le croisement est calqué sur le phénomène naturel de «crossing-over» qui agit sur les chromosomes lors de la reproduction des êtres vivants [Figure 118]. Cette opération permet un partage des caractéristiques génétiques des individus par échange de fragments de la chaîne chromosomique afin de créer le chromosome du nouvel individu. Ainsi, le nouvel individu est constitué de caractéristiques issues du patrimoine génétique de ses deux parents. Elle est donc essentielle et différencie en fait les systèmes évolutionnistes des méthodes stochastiques de type «recuit simulé».

Les systèmes évolutionnistes utilisent le plus souvent une méthode de croisement simple adaptée à la représentation choisie pour les codes génétiques. Ce croisement se fait par échange de morceaux de code génétique articulé autour d'un point de croisement. Le résultat du croisement de deux parents fournit donc deux nouveaux individus.

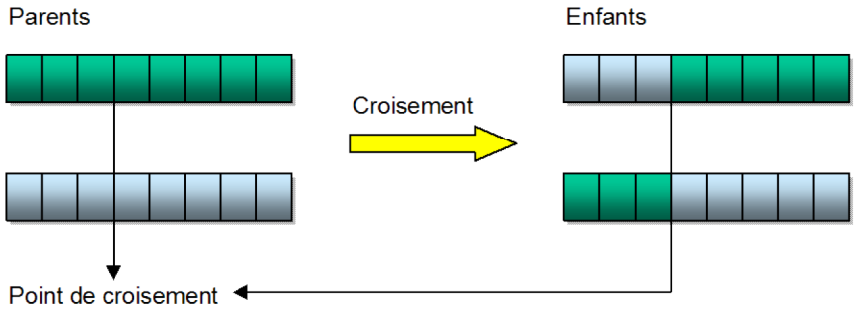


Figure 118 : *Croisement simple de deux chromosomes*

Ces individus sont alors placés dans la nouvelle population. Comme on peut le remarquer, les deux individus n'introduisent pas de nouveaux caractères mais réutilisent simplement le patrimoine fourni par leurs parents.

Le croisement intervient généralement sur la population avec un taux variant entre 50% et 90% de la population selon le problème considéré, un fort taux de croisement indiquant un fort «regroupement» des individus dans l'espace des solutions et par là même une plus grande stabilité de l'algorithme.

Pour améliorer la convergence, il existe d'autres méthodes de croisement telles que le croisement uniforme [Goldberg89] qui permet d'utiliser plusieurs points de croisement [Figure 119].

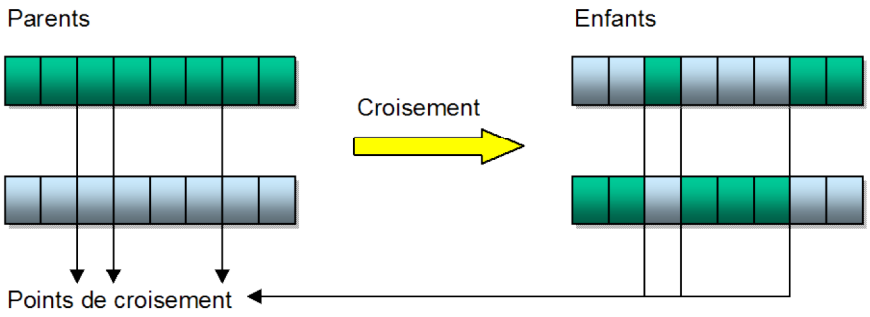


Figure 119 : *Croisement uniforme de deux chromosomes*

Il est de même possible de définir des croisements qui fournissent *n* nouveaux individus à partir de *m* parents avec *v* points de croisement.

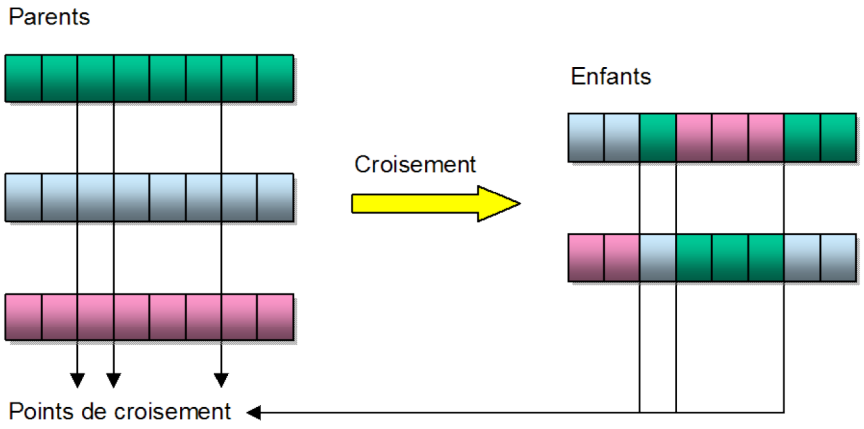


Figure 120 : *Croisement multiple de plusieurs chromosomes*

Delahaye propose un mode de sélection utilisant une technique issue du recuit simulé. Les deux parents sont alors remplacés par leurs enfants, suivant une loi de probabilité évoluant avec le temps, dans le cas où ces derniers sont plus performants après évaluation.

La mutation : la mutation va simuler le phénomène de modification intervenant sur les chaînes chromosomiques sous l'effet des agents extérieurs tels que les rayonnements ou la pollution. Elle va agir en modifiant aléatoirement une partie de l'information contenue dans la structure de donnée du chromosome [Figure 121]. Elle permettra donc à un individu d'aller aléatoirement explorer une nouvelle partie de l'espace des solutions.

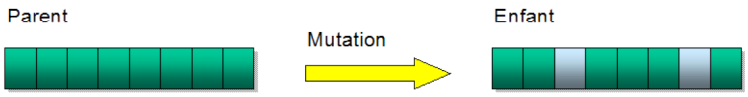


Figure 121 : *Mutation d'un chromosome*

La mutation intervient généralement sur la population avec un taux de 0,5% à 2%. En effet, un taux trop important éparpillerait la population et ne conserverait pas suffisamment le patrimoine génétique des chromosomes d'une génération à l'autre. Inversement, un taux nul ne permettrait pas la création d'un nouveau patrimoine ou la régénération de gènes perdus, pouvant être utiles à la solution recherchée. Ces gènes peuvent avoir été éliminés au début de la convergence de l'algorithme car ils ne constituaient pas une information «vitale» à ce point de la convergence.

Le réglage des différents taux d'application des opérateurs de modification de la population vont conditionner la convergence de l'algorithme. La mise en place d'une optimisation va donc faire intervenir la mise au point d'une multitude de paramètres pour que la convergence puisse s'opérer dans des temps raisonnables. De plus il est à noter qu'un paramétrage inadéquat mène généralement à la convergence vers une solution sous-optimale.

6.2 Application à la génération de comportements

Dans ce chapitre nous allons décrire différents travaux utilisant les systèmes évolutionnistes pour la génération de comportements. La plupart de ces travaux, issus de domaines afférents à la synthèse d'images (notamment robotique et biologie), possèdent néanmoins une problématique similaire : la définition de comportements cohérents pour des acteurs de simulations articulés.

6.2.1 Les travaux de Ngo

Les contraintes spatio-temporelles sont un moyen de définition de l'animation dans lequel l'opérateur propose une suite d'actions et de contraintes à respecter sans fournir le moyen de réaliser le processus complet. Cette méthode attractive de simulation ne dispose néanmoins que de peu d'algorithmes à même de reconstituer la séquence complète en prenant en compte l'ensemble des contraintes imposées au mouvement.

Ngo propose d'utiliser, dans [Ngo93], les algorithmes génétiques pour animer les squelettes entre deux situations prédéfinies. Il s'appuie pour cela sur un simulateur physique utilisant les notions de stimulus arrivant sur l'entité et de réponses par déformation de ces entités. Il applique sa technique à la recherche de mouvements réalistes de marche et de sauts en deux dimensions.

Avec ce modèle de simulation, c'est donc l'individu considéré qui produit lui-même ses déformations en fonction d'éléments en provenance de ses capteurs. Ceux-ci correspondent aux «sens» que possède l'entité. On trouve ainsi les capteurs suivants :

Jointure : la valeur des angles entre les différentes parties de l'entité.

Tactiles : les forces exercées par les extrémités sur l'environnement.

Cinétiques : les variations de vitesse du centre de masse.

Position : la position du centre de masse.

Les stimuli sont interprétés et intégrés pour fournir de nouvelles valeurs à partir de l'acquisition réelle des capteurs, à la manière des «filtres» de Terzopoulos. Ngo définit ainsi des fonctions stimulus de la forme :

$$Stimulus(v_1,\ldots,v_n) = \left(\sum_{j=1}^{n} \log\left(\frac{\lambda_j}{\lambda_j^{\min}}\right)\right)\left\{1 - \max_{j=1}^{n}\left(\lambda_j\left(v_j - v_j^0\right)\right)^2\right\}$$

Dans cette équation, les valeurs $\{v_1,\ldots,v_n\}$ représentent les valeurs fournies par les capteurs ; les coefficients λ_j et v_j^0 sont les valeurs permettant d'ajuster les filtres d'entrée.

Cette fonction de stimulus produit une excitation qui est ensuite traitée par un simulateur physique pour produire finalement le mouvement.

Les réponses consistent en un ensemble d'équations différentielles régissant chacune la déformation d'une articulation de l'entité au cours du temps. Ces équations sont de la forme $\tau^2\ddot{\theta}_i + 2\tau\dot{\theta}_i + \left(\theta_i - \theta_i^0\right) = 0$ où τ est une constante de temps, θ_i^0 la valeur finale de l'angle et θ_i sa valeur courante. L'application de l'ensemble des réponses va donc modifier l'ensemble des angles $\{\theta_1,\ldots,\theta_n\}$ pour les amener, sans discontinuité, vers les angles $\{\theta_1^0,\ldots,\theta_n^0\}$.

L'algorithme génétique permet d'optimiser les différents paramètres d'ajustement du filtre d'entrée λ_j et v_j^0 ainsi que les angles souhaités en sortie $\{\theta_1^0,\ldots,\theta_n^0\}$ et le temps τ mis pour atteindre cette nouvelle position. On obtient ainsi la séquence de mouvements à appliquer pour arriver d'un état de départ à un état final sans spécifier de manière explicite la suite des mouvements à accomplir.

La Figure 122 présente le résultat d'une simulation portant sur la recherche d'un comportement de marche. Le système génétique permet de trouver les séquences intermédiaires (en grisé) et les positions clefs (en noir). Les deux mouvements trouvés correspondent à deux morphologies différentes pour l'entité simulée.

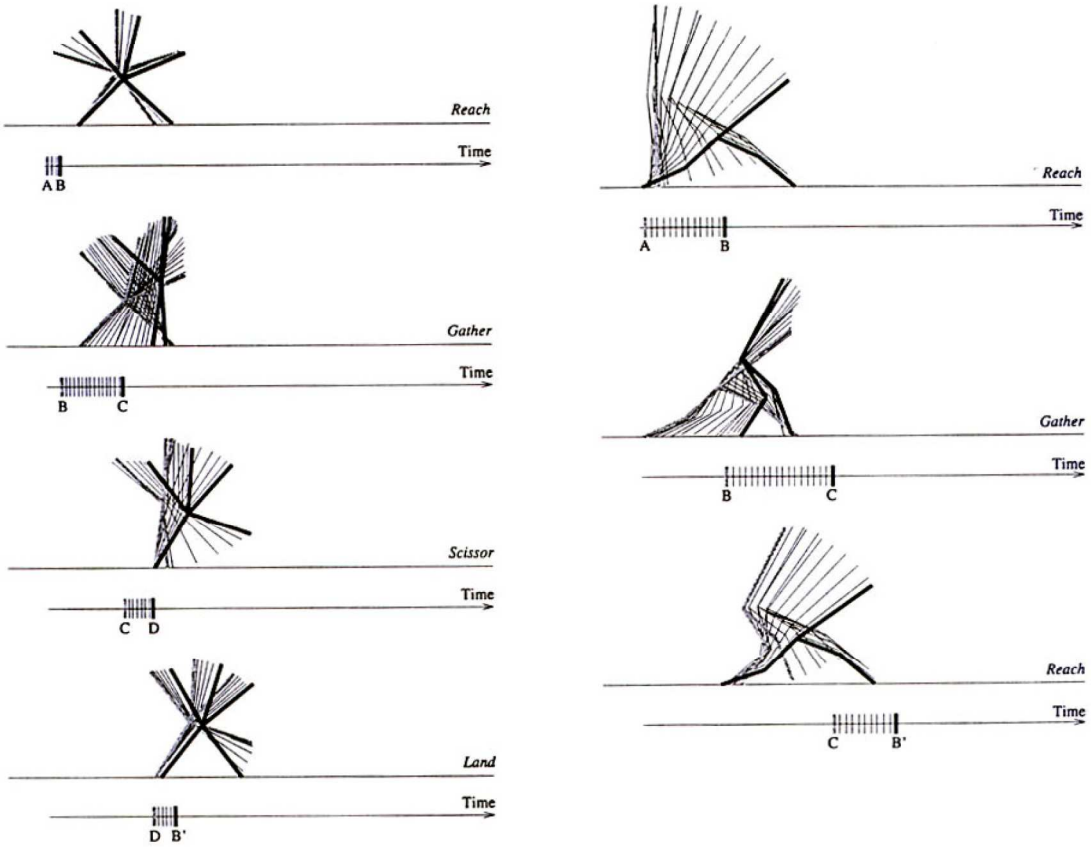


Figure 122 : *Une simulation de marche*

Cette simulation fournit des résultats très intéressants car l'animation produite semble plus «naturelle» que celles obtenues par d'autres approches, notamment celles basées sur la décroissance du gradient [Witkin88]. Son inconvénient est d'être relativement lente, interdisant son utilisation dans le cas de personnages animés en temps réel et en trois dimensions.

6.2.2 Les travaux de Gritz

Gritz a proposé une approche [Gritz95] similaire à celle de Ngo étudiée précédemment. Il propose donc un système basé sur la mécanique permettant de calculer l'ensemble des équations du mouvement permettant à un objet de passer d'une position initiale à une position finale.

Gritz utilise la programmation génétique pour générer des contrôleurs permettant à des personnages d'exécuter le mouvement souhaité.

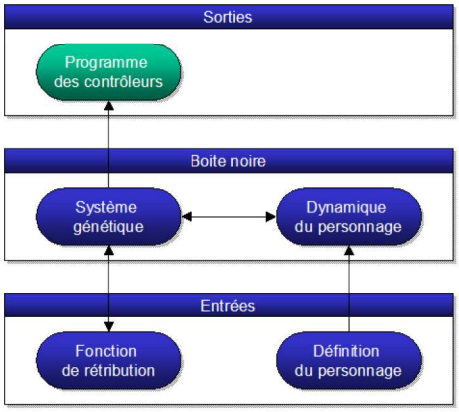


Figure 123 : *Architecture du système*

Il a donc créé un système [Figure 123] dans lequel l'utilisateur spécifie la définition (masses, liaison, contraintes) du personnage à animer et la fonction de rétribution de la trajectoire créée. Le système génétique va alors agir de pair avec une simulation de la dynamique du personnage pour produire des programmes permettant de contrôler les différents degrés de liberté du personnage. La simulation de la dynamique prenant en compte les capacités du modèle, les mouvements ne peuvent pas dépasser les amplitudes permises par ce dernier afin de toujours être plausibles [Figure 124].

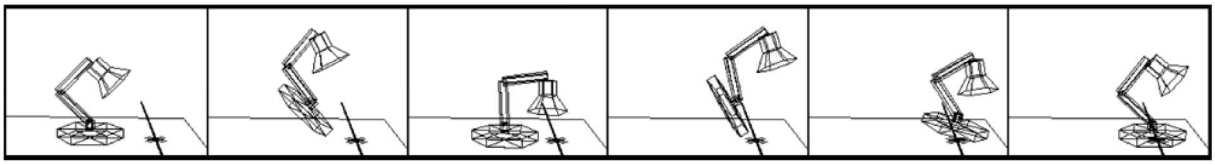


Figure 124 : *Exemple de mouvement généré par le système génétique*

Un chromosome représente l'ensemble des variables d'état de l'entité personnage telles que la vitesse, la position ou encore les forces externes appliquées. La structure de données utilisée est de type S_Expression [Koza92].

La fonction d'adaptation prend l'ensemble des données calculées par le système de simulation de la dynamique de façon à attribuer à l'individu une note correspondant aussi bien à l'objectif atteint qu'au «style» du mouvement. Cette note de style va par exemple pouvoir prendre en compte les objets heurtés, les règles de sécurité violées par le système ou encore la position de l'équilibre final.

Les résultats fournis par ce système sont très intéressants. Appliqué à un modèle classique (une lampe de travail), l'algorithme fournit des contrôleurs permettant à la lampe d'atteindre un but fixé par bonds successifs [Figure 124]. Le rajout [Gritz97] de contraintes fortes d'exclusion telles que la non intersection avec une barre présente dans la scène montre la simplicité de mise en œuvre et la validité de ce type d'approche qui génère toujours des mouvements cohérents [Figure 125] (Cette séquence a donné lieu à un film «L*xo Learns to Limbo»).

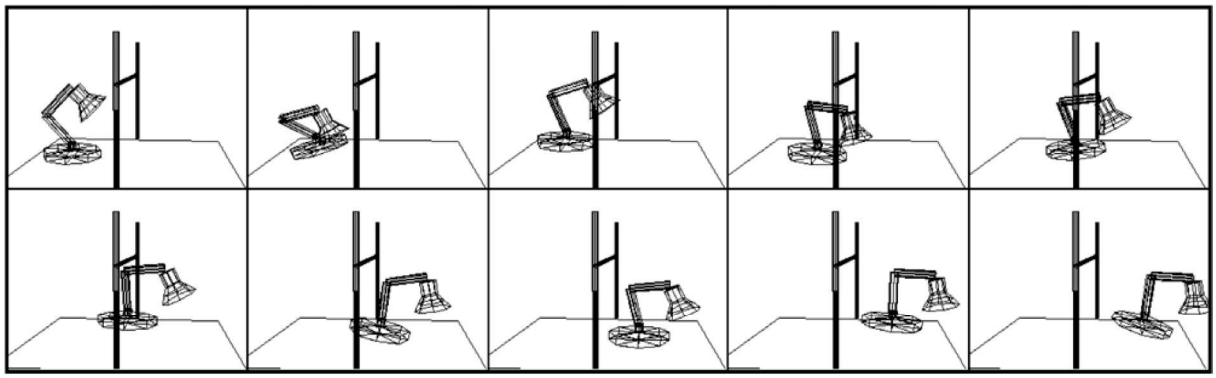


Figure 125 : *La lampe apprend à éviter les obstacles par ajout de contraintes au système*

L'inconvénient de la méthode réside là encore dans sa lenteur d'exécution interdisant son utilisation en temps réel.

6.2.3 Les approches issues de la robotique

Nous allons décrire ici divers travaux issus du domaine de la robotique et ayant trait à la génération de comportements par algorithme génétique. Ce domaine a en effet de nombreuses similitudes avec celui de la simulation comportementale dès lors qu'il s'agit de simuler le mouvement de robots afin de déterminer leurs caractéristiques de construction ou tout simplement de créer leurs organes de contrôle.

6.2.3.1 Le fil d'Ariane

Cet algorithme [Ahuactzin94] [Bessière95] permet la recherche de chemins en environnement dynamique dans l'espace des configurations d'un bras robot. Ce type de problème peut être rapproché du déplacement d'une entité dans un monde en deux dimensions.

Pour effectuer la recherche du chemin, cet algorithme utilise deux systèmes génétiques séparés. Le premier système (appelé SEARCH) va explorer l'espace des configurations et poser des jalons correspondant à des points possibles. Un second algorithme (EXPLORE) va ensuite parcourir l'ensemble de ces points pour fournir le meilleur chemin reliant le point de départ au point d'arrivée en utilisant les jalons. Cette optimisation va avoir lieu entre deux mouvements de façon à pouvoir prendre en compte les différents objets mobiles présents dans l'espace d'évolution.

L'application de cette méthode fournit de bons résultats matérialisés par le mouvement conjoint de deux bras robots. Les temps de calcul de cet algorithme étant cependant rédhibitoires pour un usage en temps réel (plusieurs minutes pour un cas simple), Bessière propose une architecture matérielle pour résoudre ce problème.

6.2.3.2 Les travaux de Cliff, Harvey et Husband

Ces roboticiens utilisent des systèmes évolutionnistes pour générer des contrôleurs de robots et des organes de liaison entre ces contrôleurs [Cliff92]. Ils utilisent pour cela un système génétique permettant d'optimiser des réseaux de neurones. Ces réseaux constituent ensuite la liaison entre l'entrée fournie par les capteurs réels et les sorties vers d'autres organes de contrôle ou vers les moteurs du robot.

Appliquée à la génération de robots utilisant des capteurs de vision réels, cette technique fournit des résultats intéressants mettant en exergue la capacité des systèmes évolutionnistes à fournir des solutions originales par rapport à celles habituellement employées.

6.2.4 Les travaux de Karl Sims

L'approche la plus aboutie en matière d'évolution de comportements et de formes reste celle de Karl Sims. Il est en effet le seul à proposer un système permettant l'évolution conjointe de comportements et de formes pour accomplir des tâches de type marche ou nage [Sims94.1] mais aussi des fonctions plus complexes telles que des tournois [Sims94.2].

Le modèle proposé par Sims s'appuie sur une analogie entre l'organe (la morphologie) et le comportement de cet organe. Il traite conjointement ces deux facettes des individus. Pour cela, il utilise un système génétique qui travaille sur des structures de données imbriquées définissant la morphologie à un premier niveau et le comportement à un second niveau. La nature étant le plus souvent constituée d'éléments de bas niveau répétés et assemblés, Sims propose une méthode récursive de définition des constituants d'une entité. Un constituant élémentaire peut ainsi être lié à lui même selon différents mécanismes permettant la répétition avec ou sans changement d'échelle de l'élément.

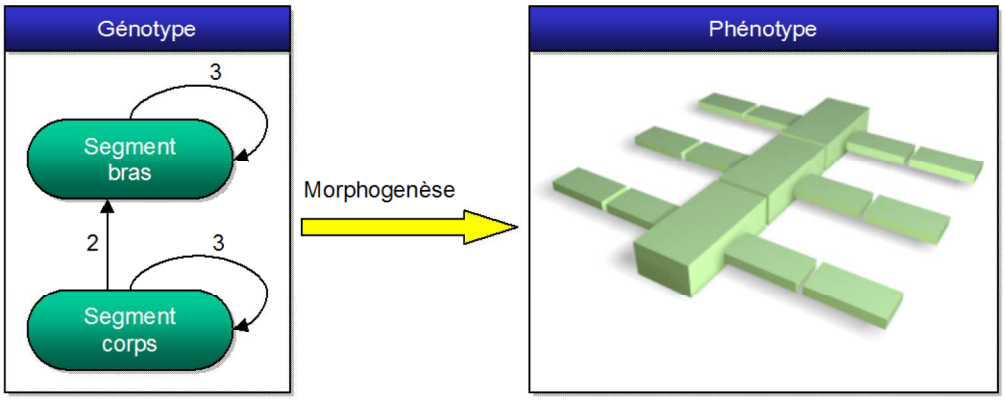


Figure 126 : *Exemple morphologie générée à partir d'une structure récursive*

Le mécanisme de liaison entre les différentes parties d'une entité permet de déterminer sa morphologie finale [Figure 126]. Chaque liaison définit ainsi la position et l'orientation relative de ses constituants. Le mécanisme de répétition est de plus indexé de façon à pouvoir générer différents niveaux de profondeur qui influenceront par la suite la morphologie de chacune des «parties» de l'entité.

Le comportement de la créature est déterminé par des réseaux de neurones situés à l'intérieur des différentes parties de sa morphologie. Les réseaux proposés par Sims sont constitués de neurones spécialisés permettant d'effectuer des opérations mathématiques. On trouve par exemple des neurones implantant les fonctions min, max, «sign-of», «smooth» ou «oscillate-wave». Chaque neurone a le nombre exact d'entrées correspondant à sa fonction. On trouve ainsi des neurones à 1, 2, 3 ou même 4 entrées. Le système prévoit toutefois un certain nombre de neurones qui ne sont présents qu'une fois dans le système et qui ne sont pas liés à la morphologie de l'entité. Ces neurones isolés, sur lesquels vont venir se connecter ceux contrôlant les structures, servent de système de contrôle global du comportement de l'entité.

La liaison entre la morphologie et le comportement s'effectue par l'utilisation de capteurs situés dans chacune des structures et d'effecteurs situés sur les liaisons. Les capteurs permettent de connaître l'état interne de l'entité (position, vitesse, angles des liaisons) ainsi que quelques valeurs issues de son état externe comme par exemple la distance et la direction d'une source de lumière. Les effecteurs peuvent faire bouger les différentes parties de la créature, indépendamment, par action sur les liaisons. C'est l'enchaînement de ces mouvements qui crée le déplacement global.

La simulation utilise les lois de la dynamique suivant le milieu dans lequel évoluent les entités. Pour la simulation de nage [Sims94.1], le milieu liquide a ainsi été simulé et les mouvements sont produits grâce aux forces d'appui des différentes parties des entités sur l'élément liquide. On voit ainsi apparaître différentes formes, plus ou moins adaptées à la nage, qui évoluent dans le temps jusqu'à disparaître ou s'adapter [Figure 127].

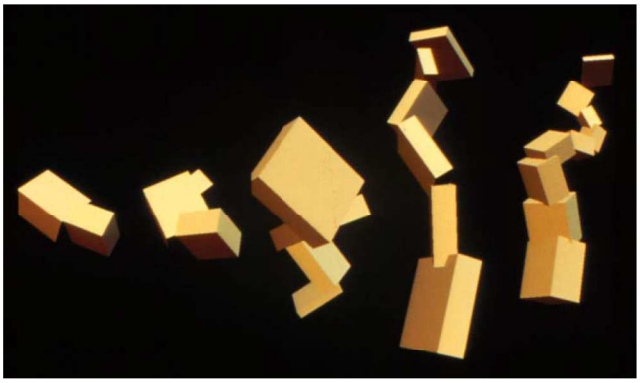


Figure 127 : *Différentes phases de l'évolution d'une entité s'adaptant à la nage*

La simulation la plus intéressante proposée par Sims [Sims94.2] consiste en la recherche de comportements d'entités qui vont lutter [Figure 128] pour s'accaparer un objet [Figure 129], le vainqueur [Figure 130] étant l'entité de gauche dans l'exemple présenté. Il utilise pour cela la co-évolution au sein d'une même population en organisant des tournois entre tous les individus de la population à une génération donnée et le meilleur individu de la génération précédente. Il obtient par ce biais une mesure réelle de l'adaptation des individus à chaque génération tout en limitant le nombre d'évaluations à réaliser.

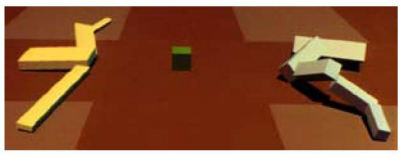


Figure 128 : *Situation de départ*

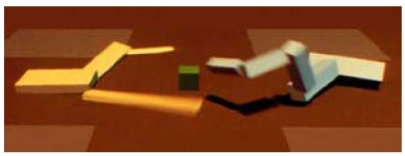


Figure 129 : *Situation de lutte*

Figure 130 : *Fin du tournoi*

L'analyse du résultat de cette simulation montre l'émergence de différents comportements [Figure 131]. On obtient ainsi des entités qui essaient d'empêcher l'adversaire d'approcher l'objectif en le protégeant. D'autres développent un «bras» très rapide qui va se déployer et envoyer l'objectif hors de portée de l'adversaire, la morphologie permettant alors à l'entité de se déplacer pour aller chercher la cible.

La fonction d'évaluation utilisée pour les deux protagonistes du tournoi prend en compte la distance finale des deux individus par rapport au cube. On utilise les fonctions d'évaluations suivantes :

$$f_1 = 1.0 + \frac{d2 - d1}{d1 + d2}$$ pour l'individu 1, d1 étant sa distance par rapport au cube

$$f_2 = 1.0 + \frac{d1 - d2}{d1 + d2}$$ pour l'individu 2, d2 étant sa distance par rapport au cube

Comme on peut le remarquer, ces fonctions permettent d'obtenir des notes entre 0 et 2.0 et d'avoir une moyenne de 1.0. En supposant que les deux entités soient à une distance équivalente du cube, leur note sera similaire et vaudra 1.0. C'est cette simple fonction de notation qui va permettre de générer une palette complexe de comportements.

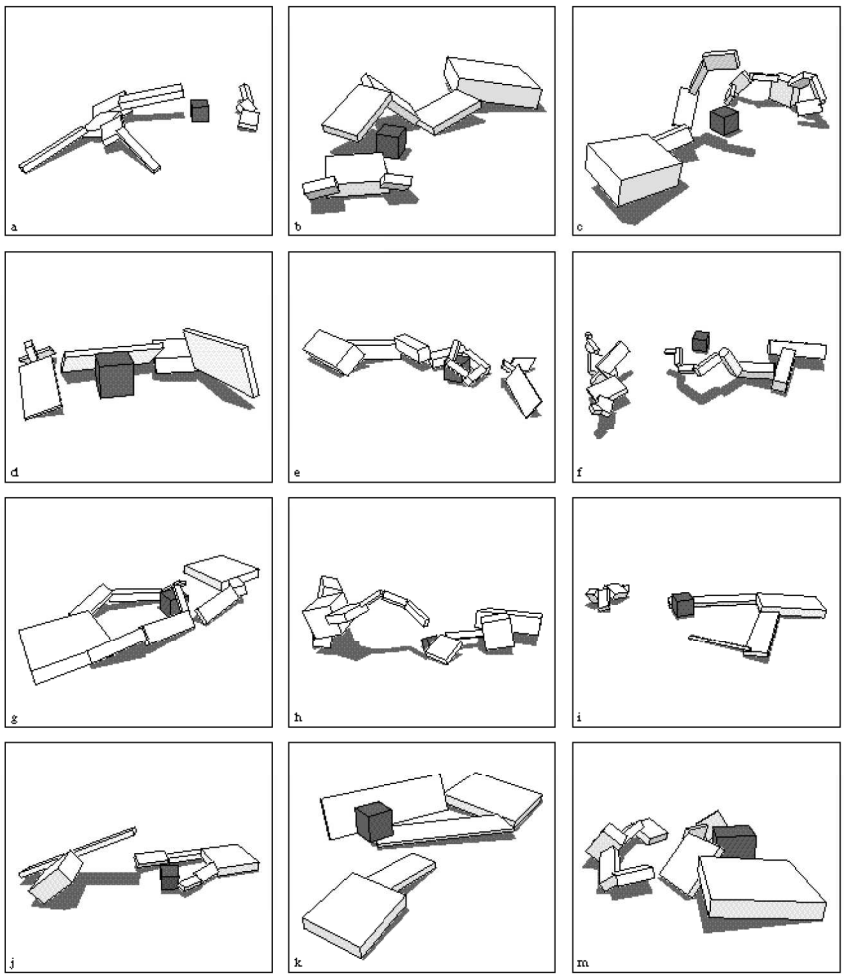


Figure 131 : *Différents types d'individus utilisant chacun une stratégie propre lors du tournoi*

En observant les comportements présentés ci-dessus on peut observer une analogie entre les résultats trouvés et certains mouvements d'espèces naturelles. Encore une fois, il semble que l'analogie existant entre évolution artificielle et évolution naturelle produise des comportements proches de ceux existant dans la nature.

Les travaux de Karl Sims constituent donc un exemple complet d'évolution en parallèle de comportements et de formes. Un des points fondamentaux de son travail réside en la modularité et en l'imbrication de ses différentes structures. Le mode de contrôle local allié à un contrôleur global permet en effet de limiter le nombre de comportements incohérents générés par le système. Néanmoins, la génération d'une entité est extrêmement coûteuse en calculs. Ceci est particulièrement vrai lorsque la fonction de notation fait intervenir plusieurs individus comme c'est le cas dans l'exemple que nous avons décrit.

6.2.5 Conclusion

L'ensemble des travaux présentés précédemment montre la validité de l'application des techniques issues de la vie artificielle pour la génération de comportements d'entités articulées. L'apport de ces modèles semble essentiel dès lors qu'il s'agit de reproduire des phénomènes naturels. Une comparaison de ces méthodes par rapport aux systèmes classiques nous montre leur supériorité lorsque la complexité recherchée se situe à la frontière d'états chaotiques, souvent difficiles à concilier en utilisant les autres techniques d'optimisation et de simulation.

Ce type de résolution de problèmes cadre bien avec la problématique des systèmes d'animation en synthèse d'images et en réalité virtuelle. On détermine souvent mieux le comportement d'un acteur d'une simulation de par les actions qu'il doit effectuer que par les moyens à mettre en œuvre pour les effectuer. Cette remarque est renforcée dans le cas où la simulation met en jeu plusieurs acteurs ou lorsque l'on recherche un comportement collectif.

Très coûteuses en calcul, ces techniques ne commencent à être exploitées réellement que depuis le début des années 90. Leur utilisation en temps réel n'a, à notre connaissance, jamais été rendue possible. Nous présentons donc, dans le paragraphe suivant, ce que nous considérons être une avancée intéressante [Balet97] [Luga98] puisque nous proposons l'utilisation des algorithmes génétiques pour gérer, en temps réel, le comportement autonome d'une entité articulée, à savoir la planification de ses mouvements ainsi que l'animation de sa structure.

6.3 Entités articulées « intelligentes »

Nous allons présenter, dans ce paragraphe, le résultat de nos travaux dans le domaine de l'utilisation d'algorithmes génétiques pour l'animation d'entités articulées en temps réel. Ces travaux ont été réalisés en collaboration avec Hervé Luga [Luga97].

6.3.1 Manipulation interactive de câbles flexibles

Nous souhaitions, avec PROVIS, pouvoir proposer au concepteur une solution [Luga98] pour manipuler intuitivement les câbles et autres objets flexibles qu'il peut être à même d'utiliser lors de la conception de prototypes virtuels. Celui-ci devait pouvoir enrouler, dérouler et déployer de tels objets de manière naturelle et réaliste. C'est pourquoi nos études se sont tout d'abord portées sur la recherche d'une solution au problème de la manipulation interactive d'une chaîne articulée de longueur quelconque, c'est ainsi que nous modélisons les câbles virtuels.

Ce problème, bien connu des roboticiens, peut être résolu à l'aide des méthodes classiques de cinématique inverse [Maciejewski90] [Zhao94] ou de dynamique inverse [Wittenburg77] [Norton92]. Ces méthodes permettent en effet de déterminer la configuration d'une chaîne articulée en fonction de la position de son extrémité. Elles sont généralement employées pour gérer le mouvement que doit effectuer un bras robot pour attraper un objet. Bien que largement utilisées dans des domaines tels que ceux de la robotique, de la biomécanique ou encore de l'animation graphique, ces méthodes posent cependant certains problèmes lorsqu'il s'agit de gérer les contraintes que peut rencontrer la chaîne articulée lors de son mouvement, celle-ci pouvant, par exemple, heurter un objet. Dans ce cas, les méthodes traditionnelles proposent généralement de bloquer le degré de liberté concerné par la collision. Cette solution n'est cependant pas intuitive en termes de programmation puisqu'elle implique la modification des mécanismes de calcul matriciel. Elle s'applique de plus a posteriori, c'est à dire après la collision, puisque rien dans ces méthodes ne permet d'éviter les contacts entre l'environnement et la chaîne manipulée. Enfin, elles ne permet pas de gérer la pression de l'environnement extérieur sur la chaîne articulée, à savoir, par exemple, la force que pourrait exercer un objet qui viendrait entrer en contact. Pour résoudre ces problèmes, le programmeur doit rajouter un module de planification qui permet de contrôler la méthode choisie de manière globale afin de respecter les contraintes imposées par la structure de la chaîne et l'environnement dans lequel elle évolue.

Nous proposons donc une solution qui permet de manipuler une chaîne articulée [Figure 132] de manière extrêmement intuitive tout en gérant, de manière simple et homogène, l'ensemble des contraintes qui lui sont appliquées.

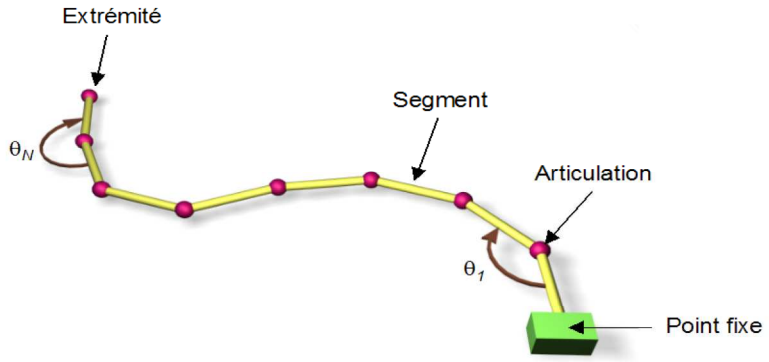


Figure 132 : *Entité de type chaîne articulée*

6.3.1.1 Mise en place du modèle

Notre objectif est donc de calculer la configuration de la chaîne articulée en fonction des modifications apportées par l'utilisateur et en respectant un ensemble de contraintes. Ces dernières peuvent représenter des forces appliquées (poids, forces exercées par l'utilisateur, rugosité ou viscosité des contacts, etc.), des zones de passage obligé, des zones à éviter ainsi que les contraintes structurelles du câble modélisé (contrainte de flexibilité locale, contrainte de flexibilité globale, etc.).

Nous disposons, pour ce faire, d'une population P d'individus représentant chacun une configuration possible de la chaîne.

Les chromosomes

Chaque individu de la population est défini par un chromosome qui doit permettre de calculer la nouvelle configuration de la chaîne à chaque pas de la simulation.

Pour cela, nous définissons la structure du chromosome de manière à permettre le calcul des différences d'angles entre la configuration actuelle et future de la chaîne articulée. Un chromosome représente donc les vitesses angulaires à appliquer en chacune des articulations de la chaîne durant l'intervalle de temps considéré.

L'expression générale d'un individu de la population est une chaîne de nombres réels de longueur égale au nombre de degrés de liberté que comporte la chaîne articulée.

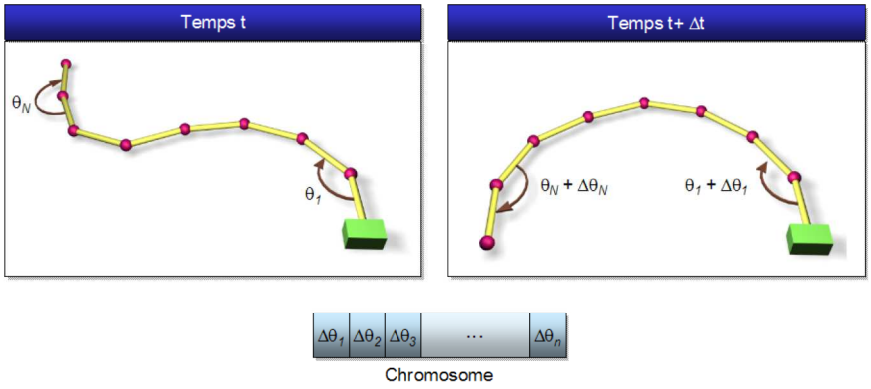


Figure 133 : *Définition d'un chromosome en utilisant les vitesses angulaires*

La configuration d'une chaîne articulée peut donc être définie à un instant t par le vecteur de réels $\left[\theta_1^t,\ldots,\theta_N^t\right]$ où θ_i^t représente l'angle courant en l'articulation i, pour $i\in[1,N]$ [Figure 133]. Notre algorithme réalise alors une fonction $F:\left[\theta_1^t,\ldots,\theta_N^t,E^t\right]\rightarrow\left[\Delta\theta_1,\ldots,\Delta\theta_N\right]$ où E^t représente l'état courant de l'environnement et $\Delta\theta_i$ la variation angulaire en l'articulation i. On obtient alors la nouvelle valeur angulaire à l'instant $t+\Delta t$ en utilisant la formule $\theta_i^{t+1}=\theta_i+\Delta\theta_i$.

En considérant que chaque $\Delta\theta_i$ peut prendre V valeurs distinctes, le nombre de configurations possibles de notre espace de recherche R peut alors être évaluée à Card(R)=V^N. En considérant, par exemple, que chaque paramètre prend des valeurs dans l'intervalle $[-0.1,0.1]$ avec une précision de 10^{-3} et que l'objet articulé comporte 20 degrés de liberté, la taille de l'espace des solutions est alors de 200^{20}, soit $1.04857\text{x}10^{46}$ configurations possibles ...

Les fonctions d'évaluation

Nous devons maintenant définir la fonction qui permettra l'évaluation de l'adaptation d'un chromosome à son environnement. Cette fonction évalue, à chaque pas de la simulation et pour chaque individu, le degré de validité de la configuration de la chaîne articulée dans le cas où l'individu traité fournirait la configuration pour le pas suivant. Elle tient pour cela compte de l'état de l'environnement et de la configuration actuelle de la chaîne articulée. C'est donc une fonction «fitness» du type :

$$fit(Chromosome)=fit(\theta_1^t,\theta_2^t,\ldots,\theta_N^t,\Delta\theta_1,\Delta\theta_2,\ldots,\Delta\theta_N,E^t)\mapsto\Re$$

Comme nous allons le voir, notre méthode permet d'arriver de manière simple à la résolution de ce problème. Nos différentes fonctions d'adaptation se situent toutes dans $\Re^-$ afin de pouvoir savoir avec certitude si la solution trouvée est optimale ou pas, une valeur de «0» correspondant alors à la valeur maximale d'adaptation et donc à une adéquation parfaite au problème.

Le premier problème présenté consiste à déterminer la position de l'extrémité de la chaîne articulée en fonction de la position du pointeur que déplace l'utilisateur. Il s'apparente donc à un suivi d'objet, la cible, dans un environnement ne comportant pas de contrainte. Dans ce cas, on définit la fonction d'évaluation à partir de la seule distance entre l'extrémité de la chaîne articulée et la cible à atteindre. On utilise pour cela la fonction d'évaluation suivante :

$$\text{fit}(Chromosome) = -\left\| \overrightarrow{cible - extrémité} \right\|$$

Si nous ajoutons au problème précédent des contraintes exclusives de non intersection avec des zones de l'environnement, la nouvelle fonction d'adaptation va devoir noter plus sévèrement les individus réalisant des intersections avec ces objets. Le problème à résoudre peut donc être défini par : « L'extrémité de la chaîne articulée doit suivre un objet cible en évitant les intersections avec les zones Z_i de la scène », la fonction d'adaptation devenant alors :

$$\text{fit}(Chromosome) = -\left\| \overrightarrow{cible - extrémité} \right\| - \eta_{collision} \cdot \sum_{Z_i} \text{nbCollisions}(chaîne, Z_i)$$

Afin de décorréler les contraintes imposées par le suivi d'objet de celles liées à l'évitement de zones, nous proposons de définir un système de fonctions d'adaptation où $\eta_{collision}$ représente le poids de la contrainte de non collision:

$$\begin{cases} \text{fit}_1(Chromosome) = -\left\| \overrightarrow{cible - extrémité} \right\| \\ \text{fit}_2(Chromosome) = -\sum_{Z_i} \text{nbCollisions}(chaîne, Z_i) \\ \text{fit}(Chromosome) = \text{fit}_1(Chromosome) + \eta_{collision}\, \text{fit}_2(Chromosome) \end{cases}$$

Nous pouvons ainsi sélectionner les chromosomes non plus en fonction d'une seule note d'adaptation mais en tenant compte de certaines de leurs aptitudes propres. Par exemple, un chromosome peut proposer une configuration de chaîne articulée qui suit parfaitement le pointeur mais en heurtant un seul obstacle. Ce chromosome recevra une très mauvaise note générale puisque l'on exclut les configurations entrant en contact avec d'autres entités. Il a même de fortes chances d'être éliminé lors de la phase de sélection suivante. Pourtant, la modification d'un seul de ces gènes aurait peut-être pu donner une configuration parfaite. C'est pourquoi la décomposition du problème en sous-problèmes, auxquels correspond, à chacun, une fonction d'évaluation, nous permet d'accélérer grandement le processus de convergence en conservant certains chromosomes ayant reçu une très mauvaise note masquant leurs aptitudes réelles.

Au problème précédent, nous pouvons rajouter des contraintes de passage par des zones préférées F_i. Ce nouveau problème peut être alors défini par : « L'extrémité de la chaîne articulée doit suivre un objet cible en évitant les intersections avec les zones Z_i de l'environnement et en passant dans les zones F_i ». Le système d'évaluation devient alors :

$$\begin{cases} \text{fit}_1(Chromosome) = -\left\| \overrightarrow{cible - extrémité} \right\| \\ \text{fit}_2(Chromosome) = -\sum_{Z_i} \text{nbCollisions}(chaîne, Z_i) \\ \text{fit}_3(Chromosome) = \sum_{F_i} \text{nbCollisions}(chaîne, F_i) \\ \text{fit}(Chromosome) = \text{fit}_1(Chromosome) + \eta\, \text{fit}_2(Chromosome) + \lambda\, \text{fit}_3(Chromosome) \end{cases}$$

Comme nous pouvons le remarquer, un des avantages majeurs de notre méthode réside en la facilité d'ajout et de classement des contraintes qu'elle autorise sans avoir à changer l'algorithme utilisé. Nous n'avons pas décrit ici la gestion des contraintes sur les degrés de liberté de la chaîne articulée. Celle-ci peut être réalisée en ajoutant, au système d'évaluation, une fonction notant chaque individu en fonction de son bon respect de ces contraintes.

Les fonctions de modification

C'est lors des phases de modification qu'entrent en jeu certains des processus d'optimisation qui permettent l'utilisation de notre algorithme en temps réel. En effet, la chaîne articulée possède plusieurs contraintes sur chacun de ses degrés de liberté. L'approche classique des systèmes évolutionnistes consiste à laisser faire « la nature » artificielle en réalisant des opérations de croisement et de mutation aléatoirement, laissant alors à l'étape de sélection le soin d'éliminer les mauvais individus.

Nous proposons d'introduire la gestion des contraintes internes lors des étapes de modification. C'est ainsi que nous réduisons grandement le cardinal de l'ensemble des solutions possibles en ne générant que des individus qui respectent, à un taux de tolérance près, les contraintes internes à l'entité. La vitesse de convergence s'en trouve grandement accélérée.

Notion de minimum local

Lorsque l'on utilise les méthodes traditionnelles pour traiter les problèmes présentés précédemment, il est possible d'obtenir des solutions correspondant à un minimum local. En d'autres termes, la méthode utilisée génère ce qu'elle croit être la solution au problème mais qui consiste, en fait, en une solution non optimale ou, pire, en une solution fausse.

Par exemple, si le problème consiste à attraper un objet sans toucher les obstacles [Figure 134], un algorithme traditionnel peut penser être sur la bonne voie car la distance entre l'extrémité de la chaîne et la cible ne cesse de décroître [Figure 135]. Pourtant, un mur infranchissable le sépare de sa cible. Il existe des méthodes (descente du gradient, backtracking) pour éviter ces phénomènes mais elles sont généralement lourdes à mettre en œuvre et ne donnent pas toujours de bon résultats.

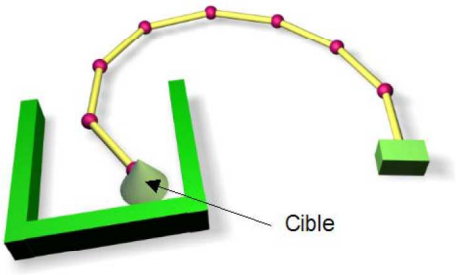


Figure 134 : *Solution correcte*

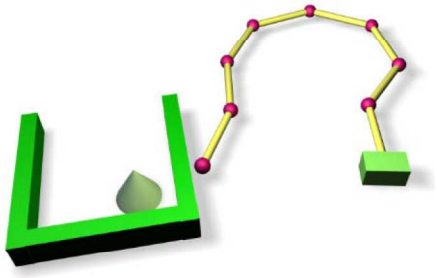


Figure 135 : *Minimum local*

Les algorithmes génétiques possèdent des mécanismes (mutations, croisements aléatoires) qui permettent d'éviter la convergence d'une population vers un minimum local. Pour améliorer encore les performances de notre algorithme, nous avons introduit une fonction de dispersion qui contrôle les mutations et choisi les individus pour le croisement en fonction de leurs positions dans l'espace des solutions. Dans l'exemple présenté [Figure 136], les individus matérialisés par des croix sont trop nombreux autour d'un des minima locaux de la fonction à optimiser. Le croisement entre ces individus produirait alors des individus explorant la même région de l'espace des solutions.

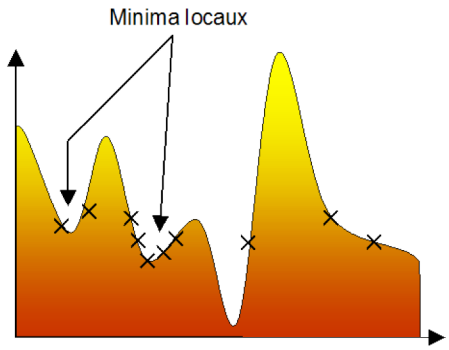


Figure 136 : *Accumulation d'individus autour d'un minimum local*

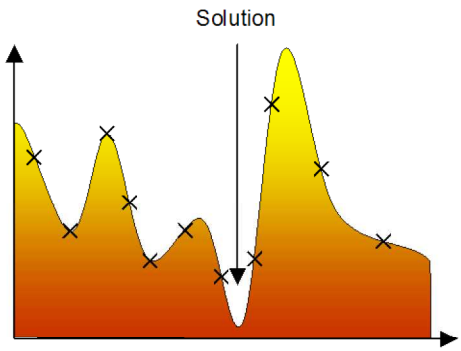


Figure 137 : *Application d'une fonction de dispersion des individus*

Notre fonction de dispersion identifie et conserve les individus ayant obtenu les meilleures notes. Les individus trop proches des meilleurs subissent des mutations et des croisements de manière à être dispersés dans l'espace des solutions. Cette technique revient à répartir ces individus de manière uniforme autour du point d'ancrage de la chaîne articulée.

Cette technique permet d'accélérer notablement les temps de convergence. Elle nécessite toutefois de pouvoir calculer la distance entre deux génotypes, ce qui est souvent difficile voire impossible lorsque le problème s'applique à des espaces où la définition d'une norme n'est pas possible.

Nous avons, de plus, mis en place un mécanisme dédié au problème traité et qui évite les emmêlements de la chaîne autour des objets de l'univers, l'empêchant ainsi de se déplier totalement. Ce mécanisme permet à la chaîne articulée d'aller explorer une autre partie de l'espace des solutions dans le cas où la cible ne pourrait être atteinte à cause de phénomènes de coinçage. Ce mécanisme de déblocage est basé sur une modification de la fonction d'évaluation lorsque l'entité articulée ne parvient pas à atteindre sa cible en un certain nombre d'itérations.

6.3.1.2 Les résultats obtenus

Nous pouvons faire une analyse du comportement de notre algorithme en observant les notes moyennes et maximales obtenues par les individus de la population au cours du temps. Les courbes présentées correspondent à la simulation d'une chaîne articulée avec 20 degrés de liberté. La fonction d'évaluation permet d'effectuer un suivi de la cible tout en évitant les obstacles. Nous avons choisi une contrainte forte sur l'évitement d'obstacles ($\eta >> \max(\left\| \overrightarrow{cible - extrémité} \right\|)$). Une note d'adaptation inférieure à -1000 traduira ainsi que l'entité articulée traverse un obstacle. Les simulations ont lieu sur 270 changements de configuration de cette dernière (avec 6 générations entre 2 déplacements) et sont extraites d'une séquence réalisée interactivement. Examinons maintenant la courbe représentant la valeur maximale d'adaptation [Figure 138].

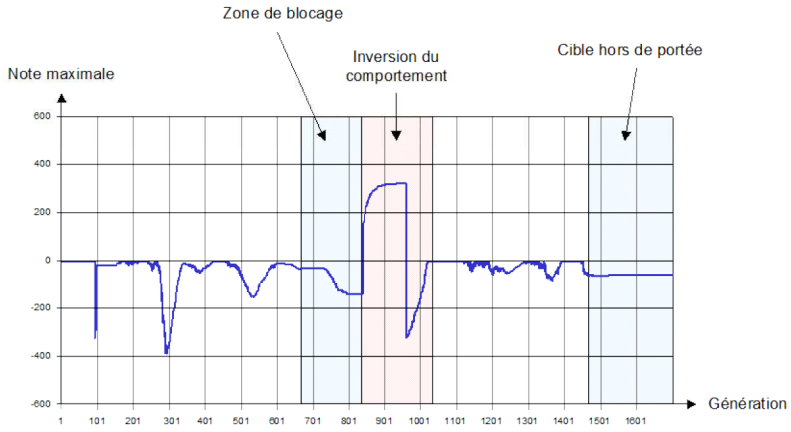


Figure 138 : *Valeur maximale de la fonction d'adaptation en fonction de la génération*

Nous pouvons d'abord remarquer que l'ensemble de la courbe se situe au dessus de la valeur critique -1000 ce qui traduit que l'ensemble des mouvements de l'entité vont s'effectuer sans collisions. De plus, la plus grande partie se situe dans une zone proche de 0 ce qui traduit que son extrémité est proche ou sur la cible. Nous pouvons isoler trois zones importantes dans cette courbe :

Zone de blocage : cette zone correspond à une incapacité de l'extrémité à atteindre sa cible à cause du blocage de l'entité entre deux obstacles.

Zone d'inversion du comportement : le comportement de la chaîne s'inverse dans cette zone, en réponse à l'incapacité précédente, afin de trouver un nouveau chemin vers la cible. Cette modification du comportement se fait par modification de la fonction d'évaluation, ce qui explique le brusque passage de la courbe dans la zone positive.

Zone d'incapacité : la cible est hors de portée de l'extrémité de l'entité qui optimise alors sa distance à la cible sans chercher à trouver un autre chemin.

Si nous examinons les valeurs prises par la moyenne des notes obtenues au cours du temps [Figure 139], nous pouvons remarquer une grande corrélation avec le comportement de la courbe précédente.

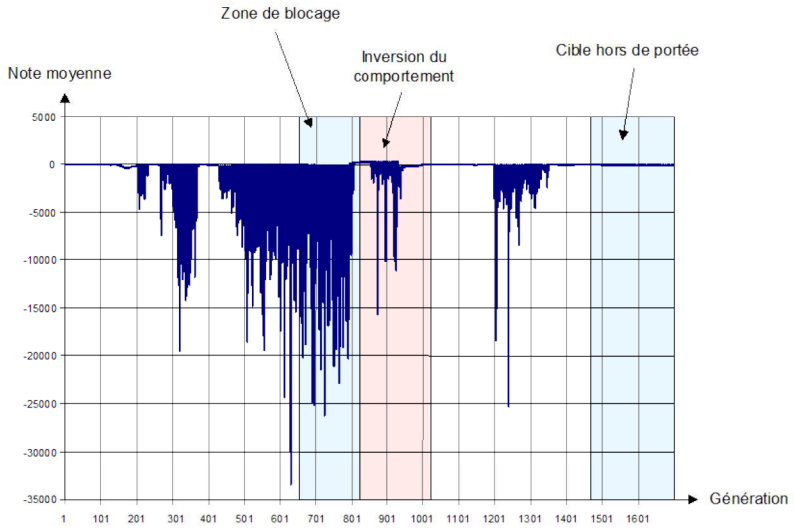


Figure 139 : *Note moyenne des individus de la population*

Nous pouvons noter que les pics correspondant à des notes moyennes basses sont de très faible durée, montrant la capacité de l'algorithme génétique à faire converger l'ensemble de la population vers des notes plus élevées. En examinant le comportement du système de recherche dans les zones stationnaires mises en évidence précédemment, on peut remarquer une rapide convergence de la moyenne de la population vers sa note maximale.

Les résultats obtenus sur différentes simulations dynamiques, tant bidimensionnelles que tridimensionnelles, montrent la faculté de notre algorithme à fournir un comportement temps réel (sur une machine de type SGI Indigo^2 R4400/250 MHz) pour une chaîne articulée comportant 20 segments avec 3 degrés de liberté entre chaque segment. Comparé à une méthode classique de type cinématique ou dynamique inverse, notre algorithme ne montre pas un effondrement des performances trop important avec l'ajout de nouveaux obstacles et de nouvelles contraintes. Il permet au contraire de programmer de manière simple et uniforme l'ensemble des contraintes applicables à une entité articulée. Il est important de noter que nous avons présenté dans ce chapitre, par souci de simplification, l'application de l'algorithme à une chaîne articulée comportant uniquement des degrés de liberté en rotation. Il est bien sur possible de mélanger différents types de liaisons (glissière, pivot, etc.) et de simuler, par exemple, le fonctionnement d'un bras robot.

6.3.1.3 Exemple d'utilisation dans le système PROVIS

Comme nous l'avons préciser en introduction de ce chapitre, notre objectif était d'étudier une méthode permettant à un concepteur de manipuler interactivement un câble [Figure 140], modélisé par une chaîne articulée, afin d'étudier son routage au sein de la maquette virtuelle. Pour ce faire, l'utilisateur dispose d'un curseur 3D qui lui permet de déplacer l'extrémité de l'entité câble dont le comportement est défini par l'algorithme décrit précédemment.

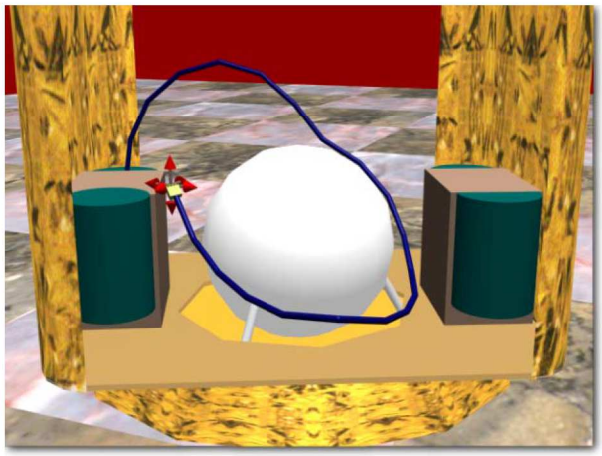


Figure 140 : *Manipulation interactive d'un câble au sein de la plate-forme d'un satellite en construction*

6.3.2 Manipulation planifiée de câbles flexibles

Nous avons présenté, dans le chapitre précédent, un mode interactif de manipulation de câbles. Cependant, il est parfois plus convivial de spécifier, dans un premier temps, un ensemble de contraintes à respecter et de laisser le système, dans un deuxième temps, résoudre le problème de manière autonome. Un exemple de ce type d'application [Figure 141] pourrait être la spécification par l'utilisateur des deux points de branchement d'un câble (source et cible), des zones par lesquelles passer (clips) et des zones de sécurité à éviter afin de ne pas passer trop près de composants critiques. L'utilisateur n'aurait alors plus qu'à exécuter la procédure de résolution et ainsi obtenir une configuration de câble conforme à ses attentes.

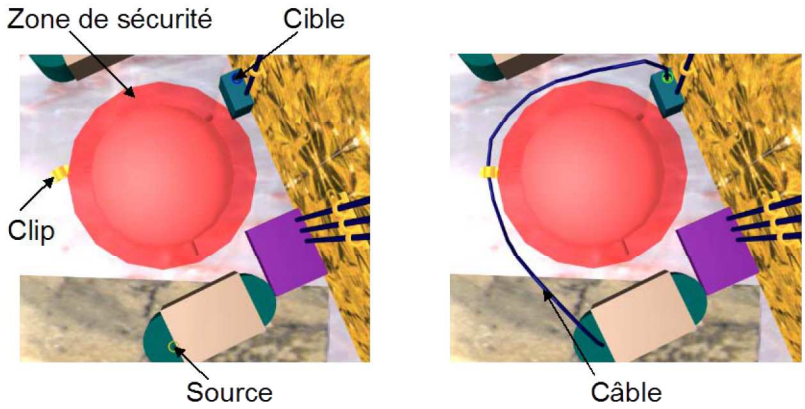


Figure 141 : *Définition des paramètres de la tâche et résultat obtenu*

Nous utilisons, pour traiter ce type de procédure, le même algorithme que celui développé pour la manipulation interactive de câbles. La seule différence réside ici en la définition des individus qui composent la population. Ils sont toujours représentés sous la forme d'une chaîne de nombre réels, de longueur égale au nombre de degrés de liberté de la chaîne articulée, mais leurs gènes représentent maintenant, non plus des vitesses angulaires, mais simplement des angles. En effet, nous ne sommes plus dans une situation où il est important de gérer l'animation du câble. L'objectif est ici de trouver la configuration respectant les contraintes prédéfinies, c'est à dire la valeur de chacun des degrés de liberté.

La configuration du câble est donc définie à un instant t par le vecteur de réels $\left[\theta_1^t,\ldots,\theta_N^t\right]$. Notre algorithme réalise une fonction F : $\left[\theta_1^t,\ldots,\theta_N^t,E^t\right]\rightarrow\left[\theta_1^{t+1},\ldots,\theta_N^{t+1}\right]$ où E^t représente l'état courant de l'environnement, θ_i^t l'angle courant de l'articulation i et θ_i^{t+1} la valeur du même angle à l'itération suivante, pour $i\in\left[1,N\right]$.

Nous présentons [Figure 142] les mesures que nous avons effectuées avec un câble composé de 20 segments possédant, chacun, trois degrés de liberté en rotation, soit un total de 60 degrés de liberté. La position des deux points de branchement ainsi que la position des obstacles et du clip étant tirées aléatoirement, nous avons mesuré le temps de résolution du problème, c'est à dire le temps que met l'algorithme à trouver une configuration correcte du câble. Les mesures ont été réitérées 10000 fois avec 1, 2, 3 et 5 obstacles afin de donner une moyenne significative du temps de résolution.

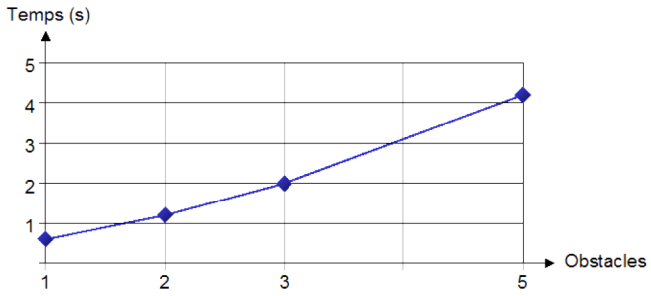


Figure 142 : *Temps moyen de résolution avec n obstacles et un clip*

Au vu des résultats obtenus, nous pouvons conclure que l'expérience menée confirme les observations faites précédemment, notre algorithme semble performant et donner de bons résultats. Forts de ses enseignements, nous avons pensé alors aborder le problème plus général de la planification de trajectoires.

6.3.3 Planification de trajectoires

La planification de trajectoires constitue un domaine de recherche très actif dans les milieux de la robotique [Taix91] [Latombe96]. Il s'agit ici de trouver la trajectoire que doit emprunter un mobile pour atteindre un objectif sans heurter d'objets. Ce problème est fondamental dès lors que l'on souhaite donner une autonomie de fonctionnement et donc de mouvement à une entité.

Il est possible de considérer une trajectoire comme étant un câble reliant une position de départ à une position d'arrivée. C'est pourquoi nous avons pensé adapter notre algorithme à ce type de problème. Il nous a fallu, pour cela, prendre en compte certaines caractéristiques propres au type de câble qui pourrait être utilisé pour ce genre de tâche. En effet, une trajectoire peut avoir une longueur inconnue puisqu'elle est fortement conditionnée par l'ensemble des obstacles qu'elle doit éviter. C'est pourquoi nous avons choisi de modéliser une trajectoire par une suite de n segments de longueur quelconque, n étant lui aussi variable.

Nous avons alors introduit, pour traiter un tel modèle, la notion de codage génétique de longueur variable. Ainsi, un chromosome peut avoir une taille variable et en changer au fur et à mesure des croisements [Figure 143] et autres mutations auxquels il est soumis.

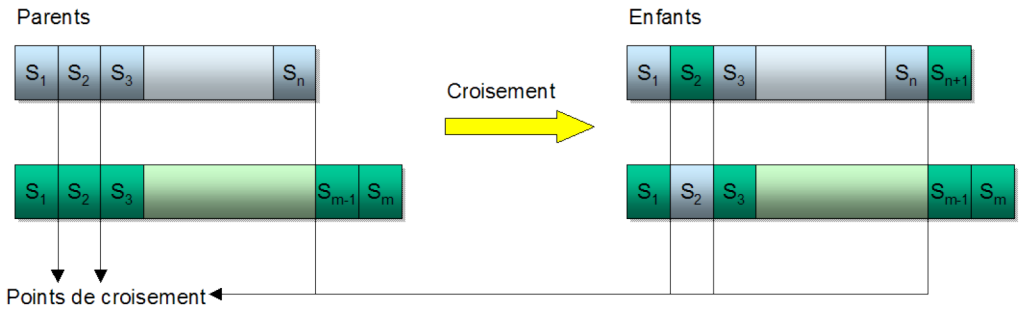


Figure 143 : *Croisement génétique de longueur variable*

De plus, le codage génétique a été étendu pour devenir hiérarchique [Figure 144] afin de pouvoir gérer deux niveaux de morphogenèse : la génération de la trajectoire en nombre de segments ainsi que la génération de chacun des segments (orientation, longueur).

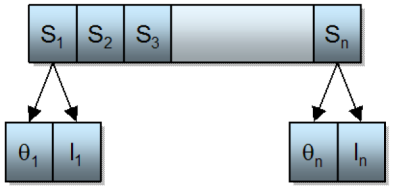


Figure 144 : *Codage génétique hiérarchique, un gène S représente 2 gènes θ et l correspondant respectivement à l'orientation et à la longueur d'un segment.*

L'implantation de ce modèle a été réalisée en modifiant légèrement le code de notre algorithme qui peut désormais réaliser les changements de longueur du code génétique d'un chromosome. Les résultats obtenus sont extrêmement prometteurs tant au niveau des performances que de la qualité des solutions fournies [Figure 145]. Le système d'évaluation contenant également une fonction qui pénalise les individus offrant les solutions les plus longues, on obtient un résultat optimal [Figure 146].

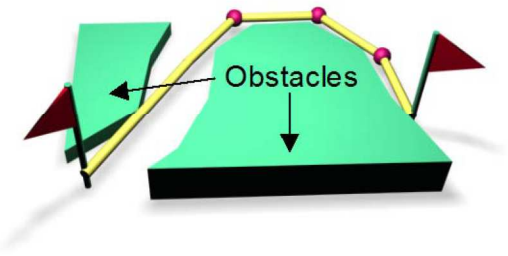


Figure 145 : *Trajectoire avec évitement d'obstacles*

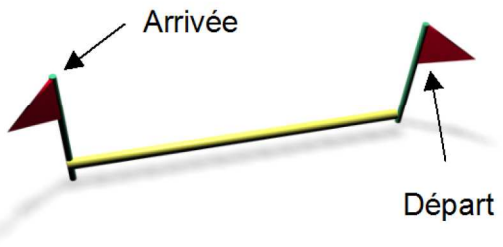


Figure 146 : *Trajectoire optimale*

6.3.4 Animation de personnages

Parallèlement à la planification de trajectoires, nous avons utilisé notre algorithme pour l'animation d'un personnage articulé en temps réel [Figure 147]. Certes ces travaux n'en sont qu'à leur début mais il nous est apparu intéressant de préciser que des résultats prometteurs avaient été obtenus. En effet, nous avons conçu une entité « personnage » qui dispose de deux modules comportementaux utilisant le même algorithme ; le comportement de cette entité est donc totalement génétique. Notre objectif est de donner une autonomie de mouvement à cette entité qui pourra se mouvoir au sein de l'environnement sans intervention extérieure. Elle dispose, pour cela, d'un planificateur de trajectoires basé sur les travaux décrits précédemment ainsi que d'un module d'animation du squelette articulé qui détermine le mouvement que le personnage doit effectuer pour progresser le long de la trajectoire choisie. Les mouvements du dos n'étant pas encore gérés dans l'exemple présenté [Figure 147], nous obtenons un personnage un peu « raide ».

L'aspect temps réel de notre algorithme nous permet de recalculer en permanence la trajectoire, autorisant ainsi les déplacements au sein d'un environnement dynamique. C'est ainsi que le personnage peut éviter, par exemple, les obstacles qui pourrait venir se placer sur sa trajectoire.

Figure 147 : *Animation, en temps réel, d'un personnage par algorithme génétique*

6.4 Conclusion

Nous avons présenté, dans ce chapitre, l'application des techniques de la vie artificielle à la génération de comportements. Nous avons démontré, expérimentalement, que l'algorithme que nous proposons pouvait être appliqué à la résolution de problèmes en temps réel, ce qui constitue, comme nous l'avons précisé précédemment, une avancé intéressante dans le domaine de la simulation comportementale.

Nous avons montré, de plus, que cet algorithme pouvait être utilisé, sans aucune modification, pour traiter différents types de problèmes comme, par exemple, la manipulation interactive de chaînes articulées, la planification de trajectoires ou encore l'animation de personnages. L'adaptation de l'algorithme à un type particulier d'application nécessite uniquement la redéfinition du système de fonctions d'évaluation qui permettent d'estimer la qualité des solutions proposées, en fonction de l'ensemble des contraintes inhérentes au problème. Comme nous l'avons vu, ces contraintes peuvent être de tous types, aussi bien locales, globales que structurelles. C'est d'ailleurs là que réside une des forces principales de la méthode proposée.

Les résultats obtenus dépassent en interactivité ceux que des méthodes traditionnelles auraient pu fournir. Il suffit, pour s'en convaincre, d'utiliser quelques minutes notre système pour mesurer avec quelle facilité il est possible de manipuler, router, enrouler ou dérouler les câbles que nous modélisons. De plus, notre modèle comportemental de personnage autonome donne déjà des résultats prometteurs. Cette voie est sans conteste à explorer plus en avant afin de pouvoir, à terme, intégrer des mannequins intelligents dans les simulations ou les outils de prototypage du futur. CISI considère d'ailleurs comme stratégique l'intégration de tels personnages au sein de ses simulateurs.

7 Conclusion et perspectives

Les travaux présentés dans ce mémoire ont été menés dans le cadre d'une étroite collaboration impliquant le laboratoire de recherche en Images de Synthèse et Réalité Virtuelle de l'IRIT, la société CISI et le Centre National d'Etudes Spatiales. C'est dans ce cadre que nous avons étudié et mis en place, pour le prototypage virtuel coopératif, une solution destinée à répondre aux nouvelles attentes des industriels impliqués dans la conception et la construction de systèmes à grande échelle.

Ces derniers sont en effet contraints de réduire toujours un peu plus les coûts de fabrication de leurs produits et ce, tout en conservant une qualité de construction irréprochable. Les phases d'avant projet et de conception, où se dessinent les grandes orientations de tels projets, sont des étapes critiques dans le processus de création d'un produit. Il s'agit là d'éviter toute erreur qui aurait alors un impact très important sur les délais et les coûts de fabrication. Paradoxalement, il n'existe, pour ces phases, que peu ou pas d'outils permettant d'appréhender et d'étudier intuitivement ce que sera le produit final.

De ces observations est née l'idée du projet de recherche PROVIS qui a pour objectif de proposer, dès les phases d'avant projet, une solution permettant de construire rapidement le prototype virtuel d'un produit. Cette solution se devait d'être suffisamment intuitive et interactive pour pouvoir être utilisée par les décideurs, peu expérimentés dans le maniement des outils traditionnels de CAO. De plus, elle devait servir de vecteur de communication au sein de l'entreprise afin de faciliter le dialogue entre les différents corps de métier très souvent dispersés géographiquement, et servir enfin de support aux méthodologies d'ingénierie concourante nouvellement mises en place.

C'est pour répondre à ces attentes que nous avons tout d'abord travaillé, en collaboration avec Patrice Torguet [Torguet98], sur le système de réalité virtuelle distribuée VIPER. Nous avons, en premier lieu, défini une architecture générale suffisamment générique pour permettre le développement de différents types d'applications. Cette architecture est basée sur le concept d'entités qui, dotées d'un comportement et munies de capteurs et d'effecteurs, peuvent communiquer entre elles à travers un réseau informatique. Nous avons ensuite proposé puis implanté un modèle de conception de comportements qui intègre les notions de niveaux de détails comportementaux nécessaires à l'optimisation des simulations temps réel. La pertinence de ce modèle a pu être depuis démontrée à de nombreuses reprises puisque CISI l'a adopté et utilisé pour le développement de ses projets de réalité virtuelle. Nous avons ensuite étudié et implanté un mécanisme permettant l'optimisation des traitements nécessaires à la détection des collisions entre entités. A ce propos, il est important de signaler que ces traitements sont une des bases fondamentales et nécessaires au développement d'un système interactif tel que PROVIS. Enfin, nous avons complété notre participation au développement de la plate-forme VIPER en évaluant et en implantant les mécanismes nécessaires à la communication audio et vidéo entre différents utilisateurs. Ce dernier point a d'ailleurs permis de mesurer la facilité avec laquelle il est possible d'intégrer de nouvelles fonctionnalités dans VIPER.

Nous disposions, à la suite de ces travaux, d'une plate-forme logicielle proposant toutes les fonctionnalités nécessaires au développement d'un système de prototypage virtuel coopératif. Il restait alors à définir l'architecture d'une telle application. Nous avons, pour cela, défini quatre types d'entités qui constituent le système PROVIS.

Nous souhaitions tout d'abord disposer d'une architecture permettant d'intégrer de manière homogène les utilisateurs dans l'environnement. Pour cela, nous avons défini les entités avatar, véritables clones des utilisateurs, qui disposent de différents modules comportementaux leur permettant, par exemple, de restituer une image tridimensionnelle de l'environnement de travail, de gérer les mécanismes de téléconférence ou de traduire un énoncé oral sous une forme textuelle. Nous avons ensuite défini l'entité Constructeur qui est en charge d'exécuter les commandes que peut lui fournir un avatar. Ce dernier peut, de plus, utiliser simultanément plusieurs modalités d'expression pour établir une commande, l'entité Constructeur étant à même de supporter le dialogue multimodal. Nous avons poursuivi nos travaux en spécifiant une entité Expert qui a pour objectif d'analyser la validité des travaux virtuels effectués et de déterminer s'ils sont conformes aux règles métier dont elle dispose. A ce sujet, nous nous sommes limités à la proposition de ce concept et à l'implantation de règles métier très simples afin d'en valider le bien fondé. Enfin, nous avons défini les entités prototypes qui sont les éléments géométriques qu'un concepteur peut manipuler et assembler afin de construire la maquette du produit final.

Nous avons de plus proposé un modèle pour la conception d'assemblages complexes dont le principe consiste, en fait, à définir un comportement d'entité à partir d'enchaînements de liaisons mécaniques. Nous avons donné comme exemple, dans ce mémoire, le mécanisme de la targette qui, sans notre modèle, nécessiterait de nombreux tests de collisions pour fonctionner.

C'est ainsi que nous avons construit le système PROVIS qui a été évalué par CISI lors de la construction de différents prototypes dont la future station spatiale internationale ou encore le simulateur de gares développé pour la SNCF. Nous nous sommes alors aperçu qu'il restait encore un domaine à aborder afin de proposer une étude relativement complète du domaine du prototypage virtuel. En effet, l'observation fut tout d'abord faite qu'un prototype virtuel ne signifie pas forcément un prototype figé. De plus, il était aussi devenu important d'étudier une solution permettant la manipulation d'entités articulées comme les différents flexibles qui composent une maquette.

Nous avons alors, en collaboration avec Hervé Luga [Luga97], tenté d'évaluer une approche originale, basée sur l'utilisation des algorithmes génétiques pour la définition de comportements pour entités articulées. L'avantage de l'approche évolutionniste réside dans sa grande simplicité de programmation et dans sa flexibilité. Un algorithme basé sur ces principes peut être, avec un minimum de modifications, appliqué à toutes sortes de problèmes. Cette approche possède cependant l'inconvénient majeur d'être très gourmande en temps de calcul. Nous avons cependant réussi, en intégrant certaines optimisations, à développer un moteur génétique qui fonctionne en temps réel et qui peut aussi bien être utilisé pour la manipulation d'entités articulées que pour la génération de trajectoires ou encore la simulation de mannequins autonomes.

En perspective de nos travaux, nous pensons qu'il serait important d'étudier comment une solution comme la notre pourrait interopérer avec un système de CAO, en charge de la définition précise des modèles pour une production directe. Comme nous l'avons précisé en introduction, il ne s'agit pas de remplacer un tel système mais plutôt de cohabiter avec, PROVIS servant de base de réflexion et de support de communication. Il serait alors intéressant d'étudier comment préserver la cohérence des modèles, modifiés dans l'un des deux systèmes, et consultés dans l'autre. De plus, l'intégration à notre système d'un dispositif à retour d'effort ainsi que l'animation de mannequins par algorithme génétique constituent de réelles perspectives pour de futurs travaux.

Enfin, il est à noter que les travaux présentés dans ce mémoire ont servi de base au projet Européen CAVALCADE soutenu par l'Union Européenne et impliquant des industriels comme SEAT, SNCF, CNES ou CSTB. Ce projet vise a développer notre système de prototypage virtuel coopératif afin de produire, in fine, une solution industrielle. Il permettra, en outre, d'évaluer sur des cas pratiques les bénéfices réellement apportés par une solution telle que celle que nous proposons.

Bibliographie

[Abarnabel96] Abarnabel R. M., Brechner E. & McNeely W., « *FlyThru the Boeing 777* », Presentation at the Digital Bayou, SIGGRAPH, 1996.

[Ahmed74] Ahmed N., Natarajan T., Rao K.R., « Discrete Cosine Transform », IEEE Transactions on Computers, Janvier 1974.

[Ahuactzin94] Ahuactzin J. M., « Le Fil d'Ariane : Une méthode de planification générale. Application à la planification automatique de trajectoires », Thèse de doctorat, LIFIA IMAG, Grenoble, 1994.

[AitAoudia94] Ait-Aoudia. S., « Modélisation géométrique par contraintes : quelques méthodes de résolution », Thèse de doctorat, Ecole nationale supérieure des mines de Saint-Étienne, 1994.

[Ames96] Ames A.L., Nadeau D.R., Moreland J.L., « The VRML Sourcebook », John Wiley, 1996.

[Andre82] Andre C., « Use of the Behavior Equivalence in Place-Transition Net Analysis », Application and Theory of Petri Nets, Springer Verlag, 1982.

[Arnaldi88] Arnaldi B., « Conception du noyau d'un système d'animation de scènes tridimensionelles intégrant les lois de la mécanique », Thèse de doctorat, Université de Rennes I, 1988.

[Astheimer94] Astheimer P. & Pöche M.L., « Level-of-detail generation and its applications in virtual reality », Proceedings of VRST'94, Virtual Reality Software and Technology, World Scientific, 1994.

[Balaguer95.1] Balaguer J.F. & Gobbetti E., « Supporting Interactive Animation using Multi-Way Constraints », Programming Paradigms in Graphics '95, Springer Verlag, 1995.

[Balaguer95.2] Balaguer J.F. & Gobbetti E., « i3D : a High Speed 3D Web Browser », Proceedings of VRML'95, 1995.

[Balaguer96.1] Balaguer J.F., « VRML for LHC Engineering », Proceedings of the Eurographics Workshop on Virtual Environments, 1996.

[Balaguer96.2] Balaguer J.F. & Gobbetti E., « 3D User Interfaces for General-Purpose 3D Animation », IEEE Computer Graphics, 29(8), 1996.

[Balet93] Balet O., « Contribution à la conception et à la réalisation d'un modeleur déclaratif multimodal », Rapport de DEA, IRIT, 1993.

[Balet94] Balet O., Gaildrat V. & Caubet R., « Les liaisons dynamiques pour un modeleur déclaratif », International Journal of CAD/CAM and Computer Graphics, Vol 9, Hermes, 1994.

[Balet96] Balet O., Torguet P., Gaildrat V. & Caubet R., « Autonomous Entities In Distributed Virtual Environments », Proceedings of Multimedia Modeling'96, Published in Towards the information super highways, World Scientific, 1996.

[Balet97] Balet O., Luga H., Duthen Y. & Caubet R., « PROVIS : A Platform For Virtual Prototyping And Maintenance Tests », Proceedings of IEEE Computer Animation'97, 1997.

[Bastide89] Bastide R., Sibertin-Blanc C., « Conception par objets de systèmes parallèles », Le Génie logiciel et ses applications, EC2, 1989.

[Bell95] Bell G., Parisi A. & Pesce M., « VRML : The Virtual Reality Modeling Language Version 1.0 Specification », 1995.

[Bessière95] Bessière P., Ahuactzin J. M., Talbi E. G. & Mazer E., « The Ariane's clew algorithm : global planning with local methods », The Algorithmic Foundations of Robotics, 1995.

[Blanchard90] Blanchard C., Burgess S., Harvill Y., Lanier J., Lasko A., Oberman M., Teitel M., « Reality Built for Two : A Virtual Reality Tool », Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics, 1990.

[Bois96] Bois J. M., Delpech M., Zilli J., Rouchouse E., « The COGNILAB / ROBOTOP Instrument: Robotic Experiment with a Force Reflecting Handcontroller During CASSIOPEE », Proceedings of ESA Workshop on Advanced Space Technologies for Robot Applications, 1996.

[Bolt80] Bolt R. A., « Put-That-There : Voice and Gesture at the Graphics Interface. », Computer Graphics 14(3), 1980.

[Bormann94] Bormann S., « Virtual Reality – Genesis and Evaluation », Addison-Wesley, 1994.

[Borning81] Borning A., « The Programming Language Aspects of Thinglab, a Constraint-Oriented Simulation Laboratory », ACM Transactions on Programming Languages and Systems, vol. 3, 1981.

[Bouzit93] Bouzit M., Richard P. & Coiffet P., « LRP Dextrous Hand Master Control System », Technical Report, Laboratoire de Robotique de Paris, Janvier, 1993.

[Brechner95] Brechner E., Helman J., Greene N., Rossignac J. & Funkhouser T.A., « Interactive Walkthrough of Large Geometric Databases », Course notes for SIGGRAPH, 1995.

[Brüderlin86] Brüderlin B., «Constructing Three-Dimensional Geometric Objects Defined by Constraints», Proceedings of Workshop on Interactive 3D Graphics, Chapel Hill, NC, 1986.

[Brison97] Brison E., « Stratégies de compréhension dans l'interaction multimodale », Thèse de doctorat, Université Paul Sabatier, Toulouse, 1997.

[Bryson93] Bryson S., Pausch R., Robinett W. & Van Dam A., « Implementing Virtual Reality », SIGGRAPH Course Notes 43, 1993.

[Burdea93] Burdea G. & Coiffet P., « La Réalité Virtuelle », Hermes, 1993.

[Buxton86] Buxton W. & Myers B., « A Study in Two-Handed Input », Proceedings of ACM SIGCHI, 1986.

[Cadoz90] Cadoz C., Lesez L. & Jean-Loup F., « Modular feedback keyboard », Computer Music Journal, 14(2), 1990.

[Calvin93] Calvin J., Dickens A., Gaines B., Metzger P., Miller D. & Owen D., « The SIMNET Virtual World Architecture », Proceedings of VRAIS'93, 1993.

[Campbell91] Campbell J. P., Tremain T. E. & Welch V. C., « The Federal Standard 1016 4800 bps CELP Voice Coder », Digital Signal Processing, Vol. 1, No. 3, Academic Press, 1991.

[Carlsson93] Carlsson C. & Hagsand O., « DIVE – a Platform for Multi-User Virtual Environments », Computer & Graphics, Vol. 17, N° 6, 1993.

[CCIR601] « CCIR Recommendation 601-2 : Encoding parameters of digital television for studios », International Telecommunication Union, 1990.

[Cerf94] Cerf R., « Asymptotic convergence of genetic algorithms », Thèse de doctorat, Université de Montpellier II - LIRMM, 1994.

[Chauvat94] Chauvat D., « Le projet Voluformes : un exemple de modélisation déclarative avec contrôle spatial », Thèse de doctorat, Nantes, 1994.

[Chevalier69] Chevalier A., « Guide du dessinateur industriel », Hachette Technique, 1969.

[Cliff92] Cliff D., Harvey I. & Husbands P., « Incremental Evolution of Neural Network Architectures for Adaptive Behaviour », Cognitive Research Papers, University of Sussex, 1992.

[Colin90] Colin C., « Modélisation déclarative de scènes à base de polyèdres élémentaires », Thèse de doctorat, Université de Rennes I, 1990.

[Conner92] Conner D.B., Snibbe S.S., Herndon K.P., Robbins D.C., Zeleznik R.C. & Van Dam A., « Three-Dimensional Widgets », Proceedings ACM SIGGRAPH Symposium on Interactive 3D Graphics, 1992.

[Coutaz94] Coutaz A & Nigay L., « Les propriétés CARE dans les interfaces multimodales », Actes des 6èmes journées sur l'ingénierie des interfaces homme-machine, IHM'94, 1994.

[David97] David P. & Guidon A. « Application de la réalité virtuelle à la SNCF », Actes du congrès Interface des Mondes Réels et Virtuels, 1997.

[Dee66] Dee J. W., « Accuracy of absolute visual distance and Size Estimation », John Wiley & Sons, 1997.

[Deering95] Deering M., « HoloSketch: A Virtual Reality Sketching/Animation Tool », Proceedings of ACM Transactions on Computer-Human Interaction, 2(3), 1995.

[Degener94] Degener J., « Digital Speech Compression », Dr. Dobb's Journal, http://www.ddj.com/ddj/1994/1994_12/degener.html, Dec. 1994.

[Deneubourg90] Deneubourg J. L., Goss S., Franks N. & Sendova Francks A., «The dynamics of collective sorting robot-like ants and ant-like robots », Proceedings of From Animals to Animats, 1990.

[Djedi91] Djedi N. E., « Modélisation en synthèse d'images : Utilisation d'une méthodologie déclarative », Thèse de doctorat, Université Paul Sabatier, Toulouse, 1991.

[Donikian93] Donikian S., « Une approche déclarative pour la création de scènes tridimensionnelles : application à la conception architecturale », Thèse de doctorat, Université de Rennes I, 1993.

[Dubois97] Dubois D., Prade H. & Yager R . R., « Fuzzy Information Engineering: A Guided Tour of Applications », John Wiley & Sons, 1997.

[Duchon96] Duchon J. & Balet O., « La réalité virtuelle, un outil de prototypage et de communication en avant-projet », Actes des 5ème journées nationales de la Réalité Virtuelle, 1996.

[Dumont90] Dumont G., « Animation de scènes tridimensionnelles : la mécanique des solides comme modèle de synthèse du mouvement », Thèse de doctorat, Université de Rennes I, 1990.

[Duthen93] Duthen Y., « Les projets VOXAR et InVitram : synthèse des travaux », Habilitation à diriger les recherches, Université Paul Sabatier, Toulouse, 1993.

[Dutoit97] Dutoit T., « An Introduction to Text-to-Speech Synthesis », Kluwer Academic Publishers, 1997.

[Eck95] Eck M., DeRose T., Duchamp T., Hoppe H., Lounsbery M. & Stuetzle W., « Multiresolution Analysis of Arbitrary Meshes », Proceedings of SIGGRAPH, 1995.

[Ellis95] Ellis S.R., « Virtual Environments and Environmental Instruments », Simulated and Virtual Realities, K. Carr & R. England, eds. Taylor & Francis, London. pp. 11-51, 1995.

[Ellis96] Ellis G., « They're Not Making 'Em Like They Used To: Virtual Reality Saves Time and Money in Manufacturing and Construction », Iris Universe, Summer, 1996.

[English67] English W.K., Engelbart D.C. & Berman M.L., « Display_Selection Techniques for Text Manipulation », IEEE Transactions on Human Factors in Electronics, Vol. HFE_8, No. 1, March 1967.

[Faugeras95] Faugeras O., Laveau S., Robert L., Csurka G. & Zeller C., « 3-D Reconstruction of Urban Scenes from Sequences of Images », Rapport de recherche INRIA n°2572, 1995.

[Foley90] Foley J.D., van Dam A., Feiner S. & Hughes J.F., « Computer Graphics, Principles and Practice », second édition, Addison-Wesley, 1990.

[Funkhouser92] Funkhouser T. A. & Sequin C. H., « Management of Large Amounts of Data in Building Walkthroughs », Proceedings of ACM SIGGRAPH, 1992.

[Genrish81] Genrish H. J. & Lautenback K., « System modelling with high-level Petri nets », Theorical Computer Science, Volume 13, 1981.

[Giard96] Giard V. & Midler C., « Management et gestion de projet: bilan et perspectives », Seconde édition de l'encyclopédie de gestion d'Economica, 1996.

[Gobbetti93] Gobbetti E. & Balaguer J.F., « VB2: A Framework for Interaction in Synthetic Worlds », Proceedings ACM UIST, 1993.

[Gobbetti95.1] Gobbetti E. & Balaguer J.F., « An Integrated Environment to Visually Construct 3D Animation ». Proceedings of ACM SIGGRAPH, 1995.

[Gobbetti95.2] Gobbetti E. & Balaguer J.F., « I3D: An Interactive System for Exploring Annotated 3D Environments », Reprinted in Scateni R (Ed) Scientific Visualization'95, Symposium Proceedings, World Scientific, 1995.

[Goldberg89] Goldberg D. E. H., « Genetic algorithms in search, optimisation and machine learning », Addison-Wesley, Reading, 1989.

[Gottshalk96] Gottschalk S., Lin M. & Manocha D., « OBB-Tree: A Hierarchical Structure for Rapid Interference Detection », Proceedings of ACM SIGGRAPH, 1996.

[Gouraud71] Gouraud H., « Continuous Shading of Curved Surfaces », IEEE Transactions on Computers, 1971.

[Greenfield96] Greenfield D., « Virtual Prototyping at Rolls-Royce », Intelligent Systems, Report 13(1), 1996.

[Gritz95] Gritz L. & Hahn J., « Genetic Programming for Articulated Figure Motion », Journal of Visualization and Computer Animation, 1995.

[Gritz97] Gritz L. & Hahn J., « Genetic Programming Evolution of Controllers for 3-D Character Animation », Proceedings of the Genetic Programming '97 Conference, 1997.

[Herndon92] Herndon K.P., Zeleznik R.C., Robbins D.C., Conner D.B., Snibbe S.S. & Van Dam A., «Interactive Shadows », Proceedings of UIST, 1992.

[Herndon94] Herndon K.P., Van Dam A. & Gleicher M., « Workshop Report: The Challenges of 3D Interaction ». SIGCHI Bulletin, October, 1994.

[Hirzinger92] Hirzinger G., Dietrich J., Gombert B., Heindl J., Landzettel K. & Schott J., « The sensory and telerobotic aspects of the space robot technology experiment ROTEX », Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space, Toulouse, 1992

[Holland75] Holland J., « Adaptation in natural and artificial systems », University of Michigan Press, Ann Harbor, 1975.

[Hoppe96] Hoppe H., « Progressive meshes », Proceedings of SIGGRAPH, 1996.

[Hudson97] Hudson T., Lin M., Cohen J., Gottschalk S. & Manocha D., « V-COLLIDE: Accelerated Collision Detection for VRML », Proceedings of VRML'97, 1997.

[Ikehara93] Ikehara C., Cole R. & Merritt J.O., « Visual-Motor Correspondence in Stereoscopic Video Displays for Teleoperated Manipulator Tasks », SPIE Stereoscopic Displays and Applications IV, Vol. 1915, 1993.

[Jain89] Jain A. K., « Fundamentals of Digital Image Processing », Prentice-Hall, 1989.

[Jensen92] Jensen K., « Coloured Petri Nets, Volume 1: Basic Concepts », Springer-Verlag, 1992.

[Kalawsky93] Kalawsky R. S., « The Science of Virtual Reality and Virtual Environments », Addison-Wesley, 1993.

[Kanawati97] Kanawati R., « Construction de collecticiels : études d'architectures logicielles et de fonction de contrôle », Thèse de doctorat, INP Grenoble, 1997.

[Kennedy93] Kennedy R. S., Lane N. E., Berbaum K. S. & Lilienthal M. G., « A simulator sickness questionnaire (SSQ): A new method for quantifying simulator sickness », International Journal of Aviation Psychology, 1993.

[Kim87] Kim W. S., Ellis S. R., Tyler M. E., Hannaford B. & Stark L., « Quantitative Evaluation of Perspective and Stereoscopic Displays in Three Axis Manual Tracking Tasks », IEEE Transportation Systems Man & Cybernetics, vol. SMC-17, 1987.

[Koenen98] Koenen R ., « Overview of the MPEG-4 Version 1 Standard », http://drogo.cselt.stet.it/mpeg/standards/mpeg-4.htm

[Koza92] Koza J. R., « Genetic Programming », MIT Press, 1992. [Koza94] Koza J. R., « Genetic Programming II », MIT Press, 1994.

[Krueger83] Krueger M., « Artificial Reality », Addison-Wesley, 1983.

[Krus97] Krus M., « Maillage Polygonaux et Niveaux de Détails », Notes et documents LIMSI N°97-10, 1997.

[Lakos95] Lakos C., « From Coloured Petri Nets to Object Petri Nets », Proceedings of the 16th International Conference on the Application and Theory of Petri Nets, Lecture Notes in Computer Science, Springer-Verlag, 1995.

[Latombe96] Latombe J. C., « Robot motion planning », Kluwer Academic Publishers, 4th edition, 1996

[Lautenbach87] Lautenbach K., « Linear Algebraic Techniques for Place/Transition Nets », Petri Nets: Central Models and Their Properties, Advances in Petri Nets'86, Lecture Notes in Computer Science 254 : , Springer-Verlag, 1987.

[Leymarie96] Leymarie F., De La Fortelle A., Koenderink J., Kappers A., Stavridi M., Van Ginneken B., Muller S., Krake S., Faugeras O., Robert L., Gauclin C., Laveau S. & Zeller C., « REALISE: Reconstruction of Reality from Image Sequences », Proceedings of the International Conference on Image Processing, 1996.

[Lewis91] Lewis J., Koved L. & Ling D., « Dialogue structures for virtual worlds », Proceedings of Computer Human Interface, ACM Press, 1991.

[Lucas89] Lucas M., Martin D., Martin Ph. & Plemenos D., « Le projet ExploFormes, quelques pas vers la modélisation déclarative de formes », Actes de GROPLAN, 1989.

[Luga96] Luga H., Destruel C., Duthen Y. & Caubet R., « Adaptive Interaction in Virtual Worlds using Evolutionary Algorithms », Proceedings of IEEE Roman'96, Robot and Human communication, 1996.

[Luga97] Luga H., « Vie artificielle et Synthèse d'images : Etude des mécanismes évolutionnistes pour la synthèse de formes et de comportements », Thèse de doctorat, Université Paul Sabatier, Toulouse, 1996.

[Luga98] Luga H., Balet O., Duthen Y. & Caubet R., « Interacting With Articulated Figures Within The PROVIS Project », Proceedings of the 11th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE), Published in Lecture Notes in Artificial Intelligence, Springer, 1998.

[Luong92] Luong Q. T., « Matrice fondamentale et calibration visuelle de l'environnement – Vers une plus grande autonomie des systèmes robotiques », Thèse de doctorat, Université de Paris-Sud, Centre d'Orsay, 1992.

[Macedonia95] Macedonia M.R., Brutzmann D.P., Zyda M.J., Pratt D.R., Barham P.T., Falby J. & Locke J., « NPSNET: A multi-player 3D virtual environment over the Internet », Proceedings ACM SIGGRAPH Symposium on Interactive 3D Graphics, 1995.

[Maciejewski90] Maciejewski A. A., « Dealing with the Ill-Conditionned Equations of Motion for Articulated Figures », IEEE Computer Graphics & Applications, May 1990.

[Martin90] Martin J.Y., « Synthèse d'images à l'aide d'automates cellulaires », Thèse de doctorat, Rennes, 1990.

[Martin93] Martin P. & Martin D., « Declarative Generation of a Family of Polyhedra », Proceedings of Graphicon'93, Saint Petersbourg, 1993.

[Meyer88] Meyer B., « Object-Oriented Software Construction », Prentice Hall, 1993.

[Meyer92] Meyer L., Applewhite H. L. & Biocca F. A., « A Survey of Position Trackers », Presence, Volume 1(2), 1992.

[Mouli93] Mouli R., Duthen Y. & Caubet R., « In VitrAm (In Vitro Animats), a behavioural simulation model », Proceedings of the 2nd IEEE International Workshop RO-MAN'93, 1993.

[Mouli94] Mouli R., « Un modèle d'animation comportementale fondé sur le concept de personnage », Thèse de doctorat, Toulouse, 1994.

[MPEG1] MPEG-1, « System Specification », ISO/IEC 11172-*, 1993. [MPEG2] MPEG-2, «System Specification », ISO/IEC DIS 13818-1, 1996.

[Nigay94] Nigay L., « Conception et modélisation logicielle des systèmes interactifs : application aux interfaces multimodales », Thèse de doctorat, Université Joseph Fourier, 1994.

[Norton92] Norton, R., « Design of Machinery », McGraw Hill, 1992.

[Ngo93] Ngo J. T. & Marks J., « Spacetime Constraints Revisited », Proceedings of SIGGRAPH, 1993.

[NSF92] NSF Research Directions in Virtual Environments, NSF Invitational Workshop, University of North Carolina at Chapel Hill, March 1992.

[Pajon95] Pajon J.L, Collenot Y., Lhomme X., Tsingos N., Sillion F., Guiloteau P., Vuyslteker P., Grillon G. & David D., « Building and Exploiting Levels of Detail: An overview and Some VRML Experiments », Proceedings of VRML'95, 1995.

[Panzic95] Pandzic I.S., Capin T.K., Magnenat-Thalmann N. & Thalmann D., « VLNET: A Networked Multimedia 3D Environment with Virtual Humans », Proceedings of MMM'95, 1995.

[Paramythioti93] Paramythioti M., « SOISIC-IPSOS3D : modélisation tridimensionnelle d'objets et d'environnements », Innocap 93, 1993.

[Paush92] Pausch R., Crea T. & Conway M., « A literature survey for virtual environments: Military flight simulator visual systems and simulator sickness », Presence, 1(3), 1992.

[Peterson81] Peterson J. L., « Petri net theory and modeling of systems », Prentice Hall, 1981.

[Petri62] Petri C. A., « Kommunication mit automaten », Thèse de doctorat, Université de Bonn, 1981.

[Pimentel95] Pimentel K. & Texeira K., « Virtual Reality: Through the new looking glass, Second Edition », Mc Graw-Hill, 1995.

[Plemenos91] Plemenos D., « Contribution à l'étude et au développement des techniques de modélisation, génération et visualisation de scènes : le projet MultiFormes », Thèse de doctorat d'état, Nantes, 1991.

[Pomello92] Pomello L., « A Survey of Equivalence Notions for Net Based Systems », Lecture Notes in Computer Science, Volume 609; Advances in Petri Nets'92, Springer-Verlag, 1992.

[Pratt94] Pratt D. R., Barham P. T., Locke J. & Zyda M. J., « Insertion of an Articulated Human into a Networked Virtual Environment », Proceedings of Simulation and Planning in High Autonomy Systems Conference, 1994.

[Rabiner78] Rabiner L.R. & Schafer R.W., « Digital Processing of Speech Signals », Prentice Hall, 1978.

[Rainjonneau92] Rainjonneau S., « Un modèle orienté objet pour la simulation comportementale », Thèse de doctorat, Université Paul Sabatier, Toulouse, 1992.

[Rechenberg73] Rechenberg I., « Evolutionsstrtegie : Optimierung technicher system nach prinzipen der biologishen evolution », Fromman Holzboog, Stuttgart, 1973.

[Reynolds87] Reynolds C. W., « Flocks, herds and schools : a ditributed behavioral model », Computer Graphics, vol. 21.4, Anaheim, 1987.

[Richard98] Richard P., Coiffet P., « Effect of Head Movements on Egocentric Depth Perception in Virtual Environments », Actes des 6éme journées de travail du GT-RV, 1998

[Robertson93] Robertson G. G., Card S. K. & Mackinlay J. D., « Information Visualization Using 3D Interactive Animation », Communications of the ACM 36(8), 1993.

[Ronfard96] Ronfard R. & Rossignac J., « Full-Range Approximation of Triangulated Polyhedra », Proceedings of Eurographics, 1996.

[Ross63] Ross D. & Rodriguez J., « Theoretical Foundations for the Computer-Aided Design System », Proceedings of AFIPS Spring Joint Computer Conference, 1963.

[Sayood96] Sayood K., « Introduction to Data Compression », Morgan Kaufmann, 1996.

[Scotton93] Scotton P., « Compression et transmission de signaux vidéo sur des réseaux haut débit avec contrôle de congestion du flux de données », Thèse de doctorat, Université de Nice, 1993.

[Segal97] Segal M., Akeley K., « The OpenGl Graphics System : A Specification (Version 1.1) », Technical report, Silicon Graphics Computer Systems, 1997.

[Sense8-98] Sense8 Corporation, « World2World : Technical overview », http://www.sense8.com/products/w2w_tech.pdf

[Shaw92] Shaw C., Jiandong L., Green M. & Sun Y., « The Decoupled Simulation Model for Virtual Reality Systems », Proceedings of SIGCHI, 1992.

[Shaw94] Shaw C. & Green M., « Two-Handed Polygonal Surface Design », Proceedings of ACM UIST, 1994.

[Sherif93] Sherif M.H., Bowker D.O., Bertocci G., Orford B.A. & Mariano G.A., « Overview and Performance of CCITT/ANSI Embedded ADPCM Algorithms », In IEEE Transactions on Communications, Vol. 41, Feb. 1993.

[Sibertin85] Sibertain-Blanc C., « High level Petri nets with data structure », Proceedings of the 6th European Workshop on Petri Nets and Applications, 1985.

[Sims94.1] Sims K., « Evolving Virtual Creatures », Proceedings of SIGGRAPH, 1994.

[Sims94.2] Sims K., « Evolving 3D Morphology and Behavior by Competition », Artificial Life IV, Brooks & Maes Editors, MIT Press, 1994.

[Slama80] Slama C. C., « Manual of Photogrammetry », American Society of Photogrammetry, fourth edition, 1980.

[Steinmetz96] Steinmetz R. & Nahrstedt K., « MULTIMEDIA Computing, Communications and Applications », Prentice Hall, 1996.

[Stone91] Stone R. J., « The UK Virtual Reality & Telepresence Project », Proceedings of Computer Graphics Conference, Vol. 1, 1991.

[Strauss93] Strauss P. S., « IRIS Inventor, a 3D graphics toolkit », Proceedings of Object-Oriented Programming Systems, Languages and Applications, ACM Press, 1993.

[Strommer93] Stommer W. M., Neugebauer J. G. & Flaig T., « Transputer-based virtual reality workstation as implemented for the example of industrial robot control », Actes du congrès Interface des Mondes Réels et Virtuels , 1993.

[Sturman94] Sturman D. J. & Zelzer D., « A Survey of Glove-base Input », IEEE Computer Graphics & Applications, January, 1994.

[Sutherland63] Sutherland I., « Sketchpad: A Man-machine Graphical Communications System », Ph.D. Thesis, MIT, 1963.

[Sutherland65] Sutherland I., « The Ultimate Display », Actes de IFIP Congress, Vol. 2, 1965.

[Taix91] Taix M., « Planification de mouvements pour robot mobile non-holonome », Thèse de doctorat, Université Paul Sabatier – LAAS, Toulouse, 1991.

[Terzopoulos94] Terzopoulos D., Xiaoyuan T. & Grzeszczuk R., « Artificial Fishes with Autonomous Locomotion, Perception, and Learning in a Simulated Physical World », In Artificial Life IV, Brooks & Maes Editeurs, MIT Press, 1994.

[Thibault94] Thibault G. & Brillaut B., « Modélisation au plus près du réel », Collection Epure de la Direction des Etudes et Recherches d'EDF, 1994.

[Thorisson92] Thorisson K. R., Koons D. B. & Bolt R. A., « Muti-modal natural dialogue », Actes de Computer Human Interface, ACM Press, 1992.

[Torguet95] Torguet P. & Caubet R., « VIPER (VIrtuality Programming EnviRonment): A virtual reality applications design platform », Proceedings of Eurographics Workshop on Virtual Environments, 1995.

[Torguet96] Torguet P., Rubio F., Gaildrat V. & Caubet R., « Multi-user interactions in the context of concurrent virtual world modelling », Proceedings of Eurographics Workshop on Virtual Environments, 1996.

[Torguet97] Torguet P., Balet O. & Caubet R., « A Software Architecture For Collaborative Virtual Prototyping », Proceedings of CompuGraphics'97, 1997.

[Torguet98] Torguet P., « VIPER : Un modèle de calcul réparti pour la gestion d'environnements virtuels », Thèse de doctorat, Université Paul Sabatier, Toulouse, 1998.

[Tremain82] Tremain T. E., « The Government Standard Linear Predictive Coding Algorithm: LPC-10 », Speech Technology Magazine, April 1982.

[Turletti93] Turletti T., « H.261 Software Codec For Videoconferencing Over The Internet », Rapport de recherche INRIA no. 1834, 1993.

[Turletti95] Turletti T., « Contrôle de transmission pour un logiciel de vidéoconférence sur l'internet », Thèse de doctorat, Université de Nice-Sophia Antipolis, 1995.

[Turner95] Turner R., « LEMAN: A System for Constructing and Animating Layered Elastic Characters », Proceedings of Computer Graphics International, Academic Press, 1995.

[Turner96] Turner R, Gobbetti E & Soboroff I., « Head-Tracked Stereo Viewing with Two-Handed 3D Interaction for Animated Character Construction », Computer Graphics Forum 15(3), Blackwell, Special Issue on Proceedings EUROGRAPHICS Conference, 1996.

[URL94] Network Working Group, « Uniform Resource Locators », IETF RFC 1738, 1994.

[Valette79] Valette R., « Analysis of Petri Nets by Stepwise Refinements », Journal of Computer and System Science, Volume 18, 1979.

[Vary89] Vary P., « Speech codec for the European mobile radio system », Proceedings of IEEE GLOBECOM'89, 1989.

[Vega García96] Vega García A., « Mécanismes de contrôle pour la transmission de l'audio sur l'Internet », Thèse de doctorat, Université de Nice-Sophia Antipolis, 1996.

[Vigouroux93] Vigouroux N., Gaildrat V., Brison E. & Balet O., « L'intégration à un niveau conceptuel », Rapport de l'atelier Interfaces Multimodales et Architectures Logicielles. Publié dans Atelier d'IHM'92, Edition ENST, 1993.

[Wallace91] Wallace G. K., « The JPEG Still Picture Compression Standard », Communications of the ACM, April 1991.

[Ware90] Ware C. & Osborne S., « Exploration and Virtual Camera Control in Virtual Three Dimensional Environments », Proceedings of ACM Workshop on Interactive 3D Graphics, 1990.

[Ware93] Ware C., Arthur K. & Booth K. S., « Fish Tank Virtual Reality », Proceedings of INTERCHI '93, 1993.

[Warnock69] Warnock J. E., « A hidden surface algorithm for computer generated half-tone », Université d'UTAH, CS Department, 1969.

[Watkins70] Watkins C. S., « A real time visible algorithm », Université d'UTAH, CSS 70101, 1970.

[Wernecke95] Wernecke J., « The Inventor Mentor : Programming Object-Oriented 3D Graphics with Open Inventor », Release 2, Addison-Wesley, 1995.

[West92] West A. J., Howard T. L. J., Hubbold R. J., Murta A. D., Snowdon D. N. & Butler D. A., «AVIARY - A Generic Virtual Reality Interface for Real Applications », Virtual Reality Systems, sponsored by the British Computer Society, 1992.

[Wilhelms90] Wilhelms J., Skinner R., « A notion for interactive behavioral animation control », IEEE Computer Graphics and Applications, 10(3), 1990.

[Witkin88] Witkin A. & Kass M., « Spacetime Constraints », Proceedings of ACM SIGGRAPH, 1990.

[Wittenburg77] Wittenburg J., « Dynamics of Systems of Rigid Bodies », Teuber Verlag, 1977.

[Yankee95] Yankee Group, « Communication, Collaboration, Coordination: The Three Cs of Workgroup Computing », Yankee Watch, Vol. 3, No. , 1995.

[Zeleznik91] Zeleznik R. C., Conner D. B., Wloka M. M., Aliaga D. G., Wang N. T., Hubbard P. M., Knepp B., Kaufman H., Hughes J. F. & Van Dam A., « An Object-Oriented Framework for the Integration of Interactive Animation Techniques », Proceedings of ACM SIGGRAPH, 1991.

[Zhang96] Zhang Z. & Schenk V., « Improvement of Online Calibration Methods: Self-maintaining calibration over time », Rapport de recherche n°2H5065/CALVIN/TN/WPB3-T3-001, ESA Calvin Project, 1996.

[Zhao94] Zhao J. & Badler N., "Inverse kinematics positioning using non-linear programming for highly articulated figures", ACM transactions on Graphics, October 1994